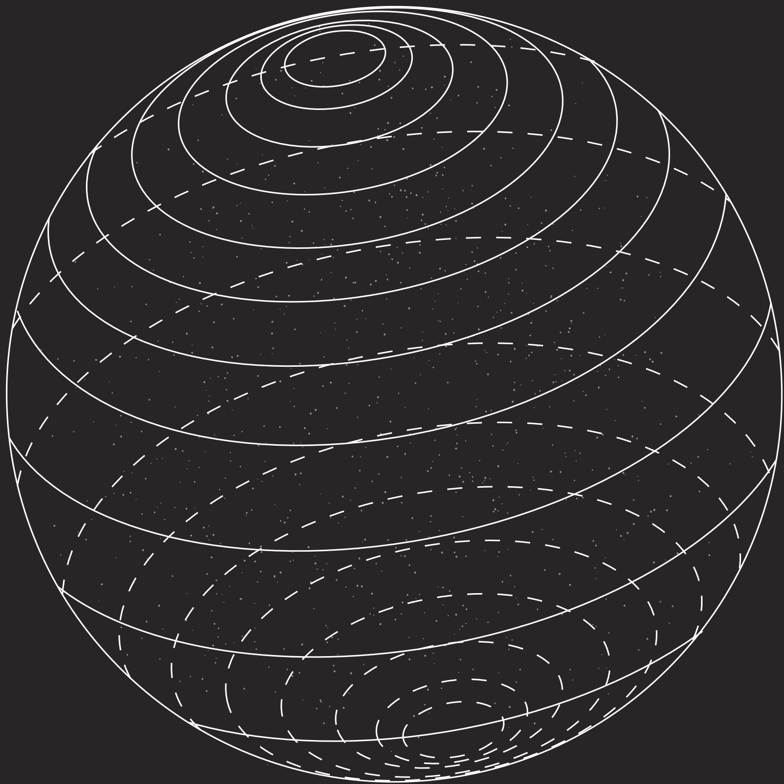


In memory of Daniel Ayre

EDITOR'S LETTER

I find myself experiencing nostalgia for a time that I was never a part of. It's normal, perhaps, for a young adult to romanticize a certain window in history that isn't possible to live today. Someone out there wishes they could be mediaeval knight while others might want to be cowboys or explorers. Others, like me, have visions more subdued. Either way, your imagination takes over and you can almost feel yourself there.

I can hear the electro-mechanical ring of heavy office phones, and papers rustling near an oscillating desk fan. I can smell the stale cigarette smoke that seems to hang in the air, the pencil eraser shavings, the burnt, luke-warm coffee resting in the break room. I can feel the grain of the dark wood paneling on the wall, the scratchy synthetic polyester of a burnt-orange-colored chair, and the cool, dull steel of a work bench.

I'm there, but I'm not. I never really will be.

Research centers, these unassuming monuments of brick and glass built to convert raw talent and bottomless capital into longshots of innovation, have mostly fallen by the wayside. Sure, there are some still out there if you look hard enough, but times have changed. The era of big research centers has passed us by, ushering in a time of startups and side hustles.

These are the places I romanticize.

I think my fascination started with Bell Labs, a research laboratory originally owned by AT&T that still survives today under the leadership of Nokia. Bell Labs has its origins in the Volta Bureau, founded in 1893 in Washington, D.C. by Alexander Graham Bell himself after receiving the Volta Prize from the French government for his invention of the telephone.

The Volta Bureau was constructed as a research laboratory, focusing on the recording and transmission of sound. By 1925, Bell Labs was formed in New York City under split ownership of AT&T and Western Electric, the engineering and manufacturing company that supplied AT&T with hardware.

Bell Labs started as a fairly safe bet for AT&T. The laboratories would work on researching new phone machinery and switching technologies to support the growth and expansion of the budding telephone network. As time went on, the number of scientists employed at the laboratory rose and their work became more varied. Over the next several decades, Bell Labs would become responsible for numerous inventions such as radio astronomy, the laser, the C and C++ programming languages, the Unix operating system, the photovoltaic cell, the one-time pad cipher, and most notably the transistor.

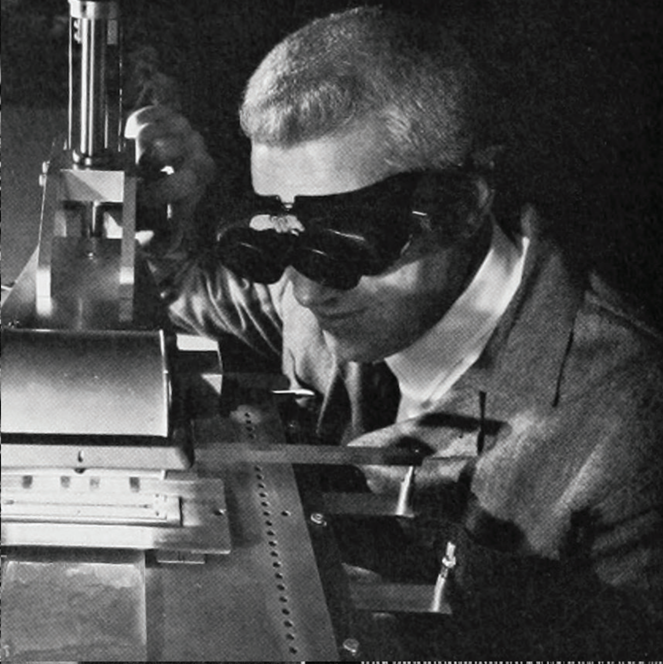
By the 1940s, other Bell Labs locations were opened outside of New York City, including offices in New Jersey, Pennsylvania, Illinois, Indiana, Ohio, Massachusetts, North Carolina, and Colorado. Over 20 offices opened exclusively for Bell Labs employees to work at, with a location in Murray Hill, New Jersey becoming the new headquarters in 1967 (which still serves as the headquarters today).

While the Murray Hill campus dripped with not-so-subtle innovation-inspired features such as an apple tree grown from a cutting of the very tree that inspired Sir Issac Newton with his discovery of gravity, the crowning jewel of Bell Labs was the Holmdel Complex. Opening in 1962, Holmdel was a corporate campus before corporate campuses. The 473-acre site could hold over 6,000 researchers and engineers within its two million square feet of working space. Designed by architect Eero Saarinen, the complex embodied a 1950s utopian outlook with mirrored curtain-wall glass and an elliptical shape. A large, 70-foot-high atrium connected four separate pavilions with sky-bridges. Small, symbolic touches promoted a sense of company pride, such as a water tower designed to look like a transistor and a sculpture of Karl Guthe Jansky's radio astronomy antenna positioned at the location the original antenna had sat in 1932 (when an older building occupied the site).

Holmdel was modern and comfortable. It encouraged collaboration and attracted genius. Seven of Bell Labs' nine Nobel Prize winning scientists and researchers worked at Holmdel over the site's lifetime, including Arno Penzias and Robert Wilson who are credited with proving the Big Bang Theory, and Steven



The Holmdel Complex



Chu who used laser light to trap atoms. By 2006 Alcatel-Lucent, then-parent of Bell Labs, sold the facility to real estate developers as it reduced its laboratory footprint. Currently, Bell Labs operates under Nokia after their purchase of Alcatel-Lucent in 2016, and boasts a total of 17 laboratories worldwide including locations in Munich, Budapest, Dublin, Cambridge, Shanghai, and Tel Aviv.

Bell Labs isn't the only research laboratory that caught my interest. Xerox PARC (the Palo Alto Research Center), a subsidiary of the photocopy-pioneering Xerox Corporation, formed in 1970 to focus on the burgeoning field of computer technology. PARC's brutalist building sat in the heart of Silicon Valley, some 3,000 miles away from Xerox headquarters in Rochester, New York. The distance ultimately proved to be a double-edged sword: there wasn't much oversight from Xerox, so PARC could operate with some autonomy, but Xerox had trouble putting faith in work they couldn't see or directly interact with.

PARC really thrived from an early infusion of computing talent. Lead by Bob Taylor, well known for his work on building and growing the ARPANET, several of initial PARC employees like Bill English and Bill Paxton came directly from SRI International's "oN-Line System" development team, featured in Bill English's famous "The Mother of All Demos" in 1969. Concepts from that project such as the computer mouse, two-dimensional display editing, and hypermedia not only went on to influence work at PARC, but change the course of computing as a whole. While other early employees would come directly from fields in academia, PARC was able to take advantage of its physical location on land leased from Stanford University to have a constant influx of new Stanford graduates looking for jobs. The PARC team was a hodgepodge of tinkerers, teachers, and techies, each bringing different points of view and new ideas. It didn't matter what you did in your previous occupation, you were part of the PARC family.

Throughout the 1970s and 1980s, PARC developed dozens of technologies that would become the building blocks for computing as we now know it. PARC is responsible for laser printing, object-oriented programming, computer-generated bitmap graphics,

Ethernet networking, WYSIWYG editing, very-large-scale integration (for integrated circuit design), the GUI, and (arguably) the first personal computer. While PARC was churning out wonder after wonder, much of it wasn't understood by parent Xerox. By December 1979, Steve Jobs and several engineers at the fledgling Apple Computer company visited PARC for three days to view demonstrations of and ask questions about the Xerox Alto, PARC's personal computer complete with a graphical user interface, computer mouse, and networking capabilities.

Xerox would infamously "fumble the future" (to borrow from the title of Douglas Smith & Robert Alexander's 1999 book about the company) as Apple would go on to use concepts they saw at PARC to influence work on the Macintosh project. The rest, as they say, is history. Throughout the '90s, PARC's image faded as they fell behind on the curve of groundbreaking innovation. By 2002 however, PARC was spun off into an independent wholly owned subsidiary company that still provides research and development services to companies such as GoogleX, BASF & Samsung.

I would be remiss if I didn't also mention Lockheed Martin's Skunk Works, a pseudonym for the aerospace company's Advanced Development Programs group. Founded in 1943, the Skunk Works name comes from the then-popular comic strip *Li'l Abner* where it is used as the name of a mysterious moonshine factory. Skunk Works' first official project was a jet fighter built at the request of the US Army Air Force. In what would become common practice for Skunk Works, work was started immediately without a formal contract and completed secretly with a small group of people in a limited timeframe. This first project, ultimately named XP-80, set a precedent for the Skunk Works group, allowing them to take on critical projects over the next few decades such as the U-2 and A-12 reconnaissance planes. Skunk Works became a de facto example of a diverse yet small team working to produce the best product. Engineers were judged by their raw skill, showcasing hires like Mary G. Ross, the first known Native American female engineer, who was among the initial 40 Skunk Works employees. Since the 1950's, Skunk Works has focused primarily on the construction of stealth aircraft, which they continue to build



today. The genericized term "skunk works" (and also the shortened "skunkworks") is commonly used today by organizations to designate a small internal team working on an innovative project.

When we released Vol 01 of the NODE zine, we didn't anticipate the warm reception we ended up receiving. We took a chance on something, something new and exciting, that we wanted to share with the world. We didn't know how people would react to the zine. Would it be loved or hated? Would people understand it? Would it be too technical or not technical enough?

Over the past year I've had the pleasure of hearing from many of you, even meeting a few in-person, and was often surprised by the audience the zine reached. It went beyond the amateur tinkerers to artists, doctors, and engineers. The material found the curious people: those rare souls out there who need to figure things out and aren't satisfied by what they see at face-value.

In a way, the NODE community is a lot like a research laboratory of its own. All of us, even though we may have never met, share in the NODE philosophy. We work on our own projects and collaborate with others. We take chances on things, and aren't afraid to get it wrong the first time. We persevere and carry on.

The subtitle for NODE Vol 02 is "Manifesting Reality" and through it we want to show that everyone has the power to build the world they want to see. Just as those engineers and researchers built the future by trying out new ideas and never giving up, we can leave our mark on this world and change it for the better.

You've already taken the first step.

We're ready, are you?

THE NODE DATA RUNNER WATCH

The NODE Data Runner is another modded Casio F91W project, this time going the extra mile to create something that not only looks nice, but is extremely functional. I guarantee there are no other watches like this out there.

Like the previous mods that you may have seen me do, this is based on the Casio F91W, and that's down to the fact that it's THE standard for an inexpensive digital watch, striking a good balance between availability, price, features and build quality. The low price has allowed me to break many of these beasts whilst on my journey of experimentation.

(Be aware if you intend to make your own, the cheap Casio fakes are much lower quality, plus the internal parts are shaped slightly differently depending on the manufacturer.)

The Data Runner name is a nod to the Cyberpunk genre of fiction which I'm fond of, specifically Johnny Mnemonic in this case because of the storage features I've added.

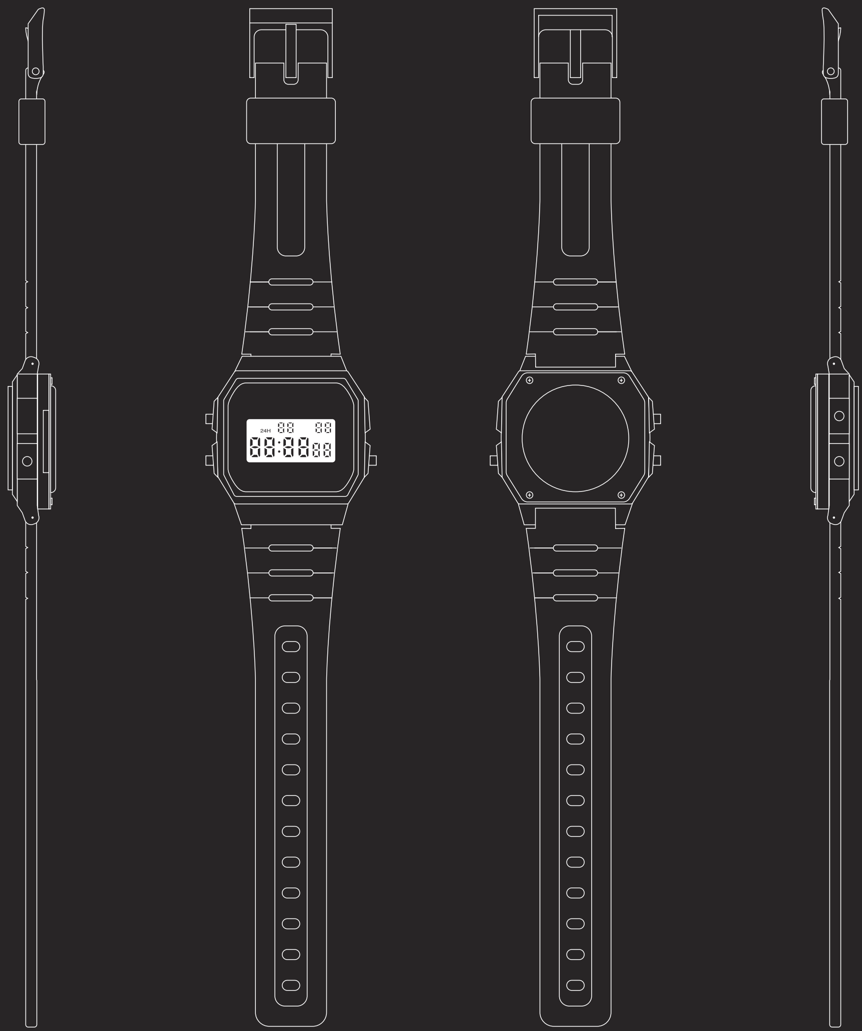
MODIFICATIONS

The first thing you'll notice is that the standard frontplate has been replaced by a minimalist black one. This is 1mm acrylic with black UV printing on the back side. I've only tried solid black, but I'm guessing you could put all sorts of designs on it using this technique.

Behind that is a custom PCB which acts as an antenna for an NFC chip. This method gives you much better scanning range, and the two contacts on the back of the antenna allow you to add any kind of NFC chip you can solder and fit inside the watch.

I chose an 8KB chip and connected it using some super thin enamel coated wires, but there are many options including tap-to-pay chips, door access, etc.

The 8KB chips are one of the largest capacity you can buy right now and these are perfect for in-person sharing of public encryption


24H


MO 39
16:25 55


8GB
CASIO
F-91W
STAINLESS STEEL BACK
WATER RESISTANT
MADE IN CHINA


24H
MO
39
16:44

keys, your resume, or other pieces of data.

Another potential upgrade would be to use a flexible PCB, which would be much thinner. In theory that'd allow you to use the original frontplate without having to remove it, for maximum stealth.

I also experimented with wiring up the top left LED light button as a way to only initiate the chip once pressed, adding an extra layer of security by stopping passive activation.

You can utilize the button motion from the Casio's custom switches, but it's very fiddly to get working. I believe you could fit a small side actuated button inside the case too, but I need to figure out a better way to implement it.

The internal green LED has now been replaced with a white one, which is purely for aesthetic reasons, because why not?

Moving to the back of the watch, you'll notice that it's slightly thicker than a regular F91W, and that's because I've added a spring-loaded micro SD card socket to the inside of the original Casio metal backplate.

It's held in place using a simple 3D-printed frame, and allows you to physically carry your important files on your person at all times. With micro SD card capacities reaching over 500GB now, that's a lot of potential data.

I designed a version with a micro USB card reader built in too, just to see how it would turn out, but the final watch was far too thick and looked a bit derpy.

All these mods give you a pretty useful little watch, with functionality and looks that you won't see anywhere else. Yes, you could use a "smart" watch, but it will spy on you and has a shit battery life in comparison.

This baby tells the time, sets alarms, stores data, can potentially pay for stuff, and has a battery that can last months. What more could you want?

THE OPEN BOOK

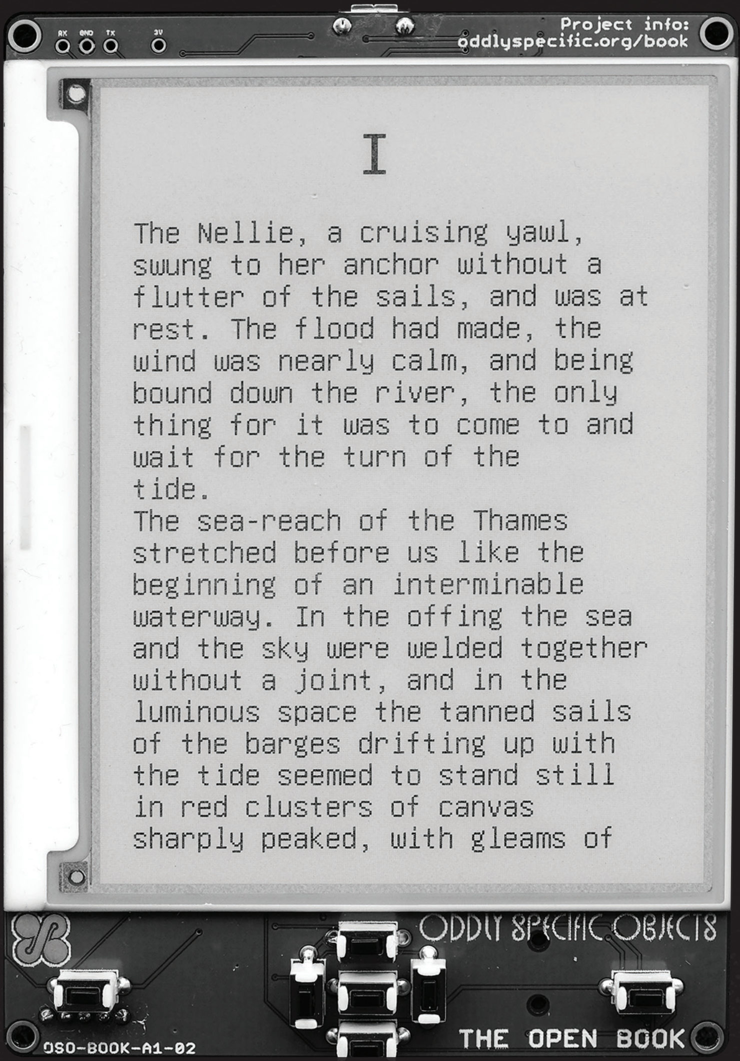
The Open Book is a universal reading device, based on the Feather-compatible SAMD51 microcontroller board. Created by Joey Castillo in early 2019, this neat little thing is shaping up to be an open source replacement to the Kindles of the world. Unlike others, this e-reader isn't locked into the ebook DRM and surveillance ecosystem.

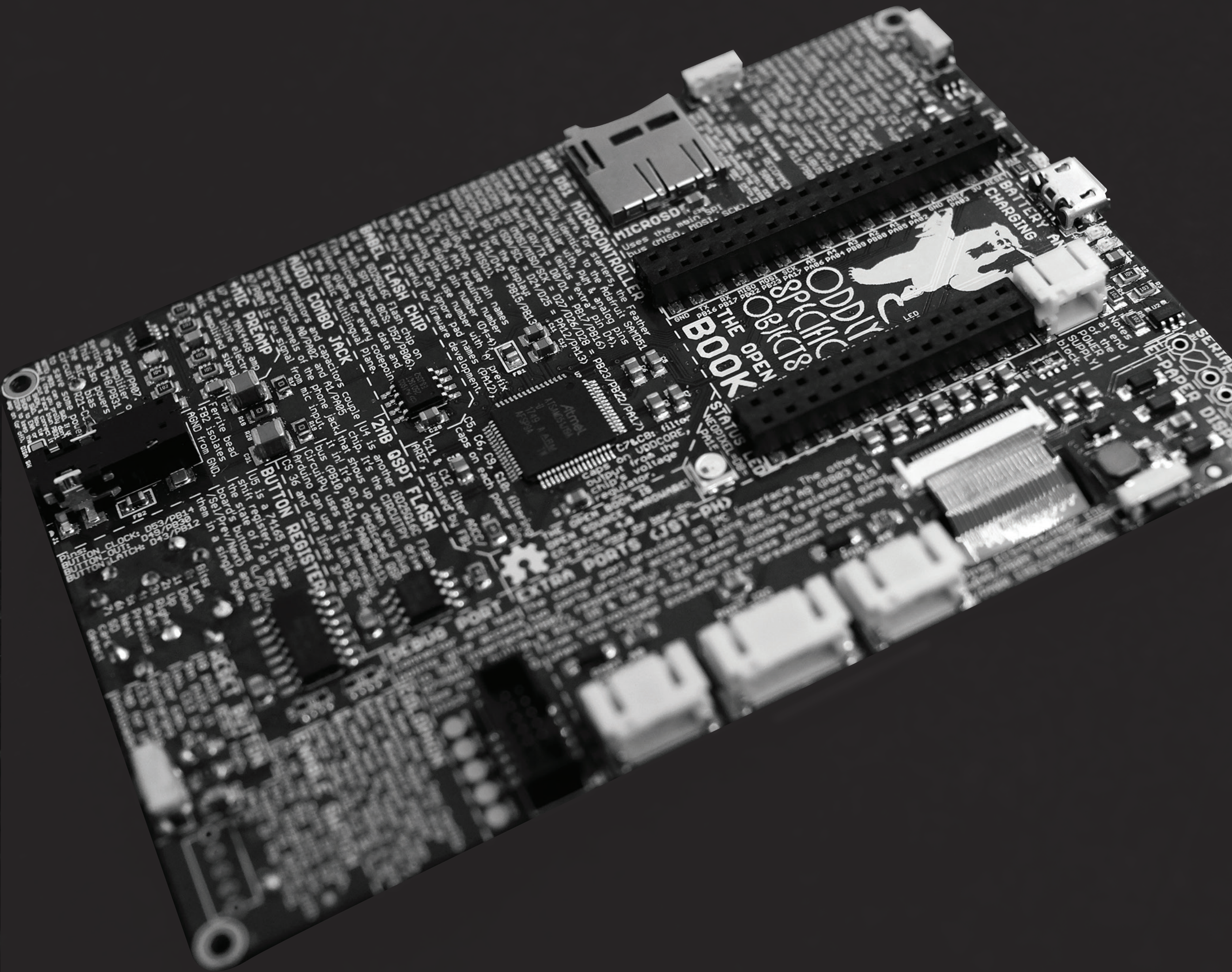
Joey explains some of the downsides of other closed systems, "There was that incident where Amazon deleted copies of *1984* from people's Kindles. And the well-known way that the Kindle surveils your reading habits. Then in June there was this tweet that kind of blew up in my feed. It was about the Microsoft eBook store closing, and one sentence just seemed mind blowing: 'The books will stop working'."

With no formal training in electronic engineering, and a long and winding resumé ranging from journalism graduate to cocktail guide, filmmaker, iOS developer, and now open hardware designer, Castillo shows why technology is so powerful. Using already available information, he is creating brilliant, useful, and important things for potentially millions of people.

Besides the need for an open e-reader alternative, another open source project was one of the main reasons why the project was started. Joey recalls, "Weirdly it all started with Unicode. I've always been fascinated by this effort to catalog and represent all the writing systems of the world, and I was doubly fascinated with GNU Unifont, an open source bitmap font that contains representations of very nearly every character in Unicode. Over 20 years, dedicated volunteers have been placing pixels so that we could have one universal font for human language. It made me want to build some kind of 'universal language' device."

He continues, "I started to think more deeply about why this thing mattered. An open source, build-it-yourself device for reading texts in all the languages of the world seems like something that ought to exist. Something you can understand from voltage levels to bits to bytes to words to feeling moved.


Project info:
oddlyspecific.org/book
RX
GND
TX
3V
I
The Nellie, a cruising yawl, swung to her anchor without a flutter of the sails, and was at rest. The flood had made, the wind was nearly calm, and being bound down the river, the only thing for it was to come to and wait for the turn of the tide.
The sea-reach of the Thames stretched before us like the beginning of an interminable waterway. In the offing the sea and the sky were welded together without a joint, and in the luminous space the tanned sails of the barges drifting up with the tide seemed to stand still in red clusters of canvas sharply peaked, with gleams of
ODDLY SPECIFIC OBJECTS
THE OPEN BOOK
OSO-BOOK-A1-02


MICROSD
MICROCONTROLLER
BATTERY
CHARGING
THE OPEN BOOK
ODDLY SPECIFIC OBJECTS
STATUS LED
MIC PREAMP
AUDIO COMBO JACK
BABEL FLASH CHIP
2MB QSPI FLASH
BUTTON REGISTER
EXTRA PORTS (JST-PH)
DEBUG PORT
E-PAPER

And, to bring it full circle, I'm hoping to make plain old UTF-8 text the main format that the Arduino library supports. It's comprehensible, it supports all the world's writing systems, and (hopefully) it won't stop working for quite some time."

The current iteration of the Open Book features a 4.2" e-paper display, with partial refresh, physical selection and page turn buttons, microSD slot for ebook data, potential for universal language support, headphone support for audiobooks, and microphone support for voice control. While the device doesn't have a backlight, a frontlight LED option is pretty easy to add to the design.

The bill of materials for V2 comes to around $60 which is pretty reasonable for something with the potential this has.

In terms of the current state of the project, Joey explains, "The hardware design is almost complete; all the core functions of the board work, and I'm in the process of working on an Arduino library to support the device. I might make one last revision of the board (removing a component to cut the BOM cost, and freeing up a couple of pins for some extra functionality), but as of now the design works."

The next steps involve making the design ready for manufacturing, and designing the casing. Then, there's a bit of work needed to polish up the software side of things. Joey envisions developing a new e-reader firmware standard, "I love the idea of building an open source e-reader firmware that can make the most of this kind of bare metal microcontroller; the Kindles and the Kobos of the world have a lot of RAM and spare cycles to play with, but if we sharpened our focus, did away with having to connect to an online storefront, didn't bother with the megabytes of code dedicated to surveilling readers as they turn the pages, we could build something lean and capable that could accomplish that baseline use case of reading books on a screen."

If you'd like to collaborate with Joey on the Open Book, and help make this a reality, check out the links below:

`hackaday.io/project/168761`
`github.com/joeycastillo/The-Open-Book`

THE IRIS PROJECT: IN CONVERSATION WITH MARTTI MALMI

Martti Malmi is a name that some early Bitcoiners may be familiar with. Being one of the very first devs working on the protocol, he got a chance to work directly with Satoshi Nakamoto, paving the way for modern peer-to-peer technologies.

A long-time proponent of decentralization, he's now concentrating his energies on Iris, a JavaScript based communications platform.

Here's my interview with Martti, where he explains the Iris project, web-of-trust systems, and the considerable goals he has for it.

Check it out. →

What motivated you to start this project?

I wanted a web of trust based alternative to all kinds of centralized databases: social media, online platforms, name services, government registries and identity providers. No single source of truth that has too much power. Instead, you could have a *web* of trusted parties—a social graph—that you can use to filter and curate what you see.

I was also interested in the natural system of trust that worked so well in tribes and village societies, but not anymore when most of the people you come across are strangers. Digital technology could be used to scale up Dunbar's number—"social scalability" as Nick Szabo puts it.

A global web of trust could be a cost-effective and less harmful alternative to central lawmakers and courts. It would be especially useful for international low-value transactions and disputes, where traditional channels are not useful.

What kind of tradeoffs did you have to make going the pure browser-based route?

Browser applications have dramatically lower barrier to entry than apps that you need to install separately. On the other hand, their p2p capabilities are limited. You can do direct WebRTC between browsers, but you still need a rendezvous server.

However, browser applications are easily ported into Electron desktop apps which can do more. Iris Electron app can already communicate with local network peers over multicast. I'll need to make it optionally join a swarm of peers that browsers can connect to.

I'm also working on a React Native mobile app. I want it to support Bluetooth mesh networking at some point. Most importantly, it will feature push notifications from messages.

The advantage of going browser-first is that all these products can use the same JavaScript codebase and easily talk to each other.

Am I right in thinking that IP addresses of the users are public at the moment? Are there any plans to add onion-routing, or some other way of increasing user privacy?

Currently it only connects to an initial list of peers and asks your friends for more public peers. Your IP address is only visible to the peers you connect to. I'm envisioning private peer exchange, so users can form "darknets" that are more difficult to censor than publicly listed peers.

You can already use the browser application over Tor without a problem. Tor support for the desktop application has been requested, and could be done using a SOCKS proxy.

I get the point—I'm not entirely comfortable sharing my IP either. It's good to have the Tor option available. But darknets might be more difficult to censor than Tor entry relays, and sometimes you don't have access to the global Internet..

Can you explain the Web of Trust system you adopted and some of its advantages?

A web of trust (WoT) consists of users who _trust_ (or more familiarly, friend) each other. Users can also distrust/block/downvote bad actors. This social graph can be used to filter out spam, fake accounts, trolls and other



unwanted content without giving the power to central moderators. You can use it to prioritize the storage of data by its author—very useful in a p2p network.

Users can also attest each others' names and other contact details. This way you build a "global address book": a name service that maps non-unique natural names to contacts in your WoT. You have a dropdown search that shows the person's name, avatar, WoT distance and whatever attributes you want. This is intuitive and avoids the name squatting problem of unique-name systems such as DNS. And it's non-hierarchical: you don't need to pay any registrar.

Have you thought about integrating Iris on mesh networks? I can imagine Iris Messenger being really useful on city-wide meshes.

Absolutely. Iris is built on Gun, a data synchronization protocol / distributed database. There are Gun adapters to sync over many kinds of transports and storages: filesystem, localStorage, websockets, webRTC, multicast and others—even the unholy AWS S3. Would be great to add Bluetooth and other mesh transports (that don't already provide TCP/IP) to the list.

Nice. How many people have used Iris so far? What kind of feedback have you received?

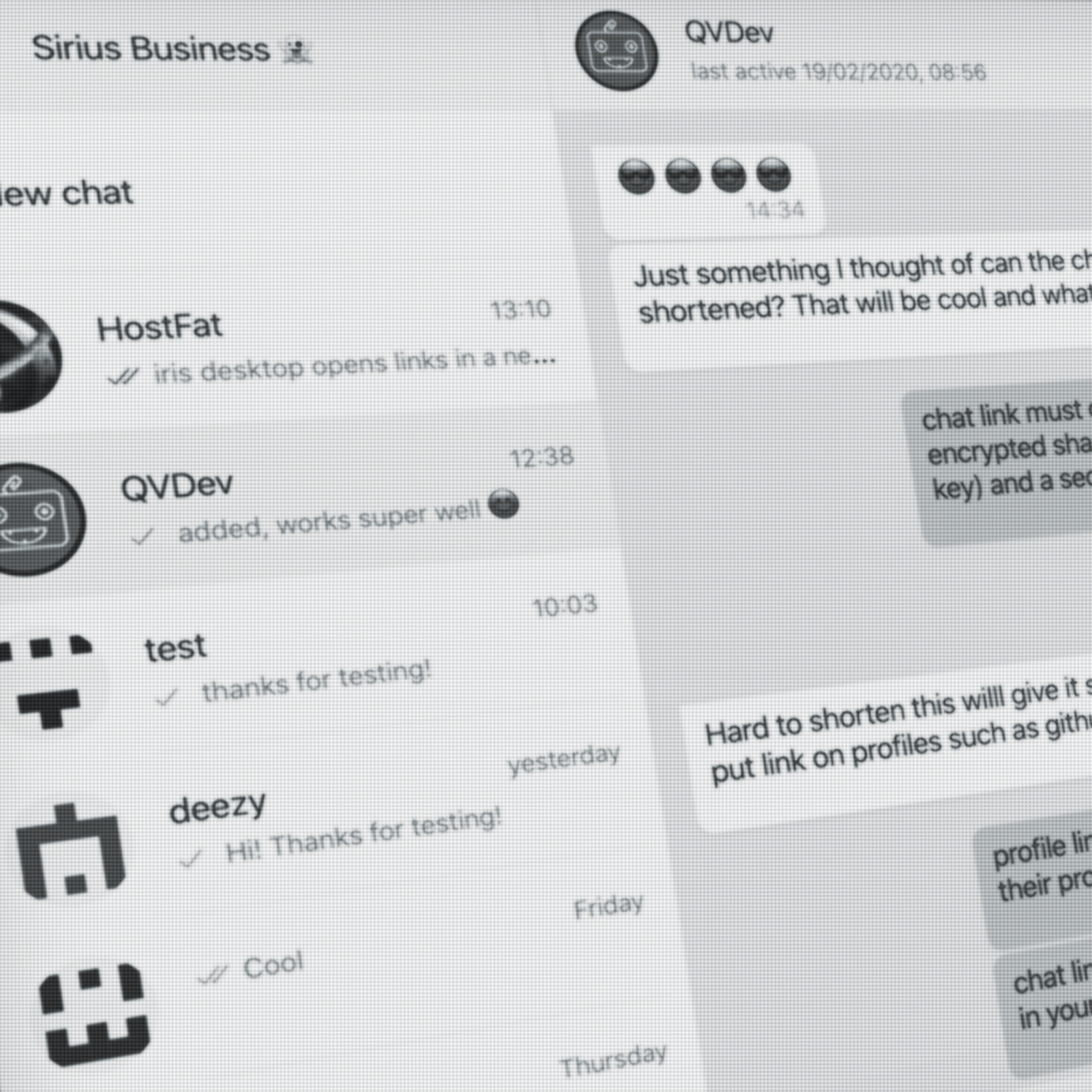
I've worked on the concepts behind Iris for years, but the most popular, functional and fun application has been Iris Messenger. It's an instant messenger which I hacked together in January 2020 using just iris-lib, jQuery and some helpers. I've had chats with many friends and strangers over it.

Feedback has been overwhelmingly positive, but there have been probably less than a few hundred people testing it.

I'll need to add group chats and mobile notifications to make it more engaging. Then I imagine decentralization projects could be the early adopters.

Is there any specific assistance you're looking for on the project?

There are lots of easy and fun improvements you could do to the browser app. Here's your


Sirius Business
QVDev
last active 19/02/2020, 08:56
ew chat
14:34
Just something I thought of can the
shortened? That will be cool and
HostFat
13:10
iris desktop opens links in a ne...
chat link must
encrypted
key) and a
QVDev
12:38
added, works super well
test
10:03
thanks for testing!
Hard to shorten this willl give it
put link on profiles such as
yesterday
deezy
Hi! Thanks for testing!
profile
their
Friday
Cool
chat
in
Thursday

chance to build the instant messenger you've always wanted. If you want more challenge, on the library level we need group chats, improvements to peer finding, privacy and other core features. Or you can just integrate the library to your own project, see how it goes, and submit any necessary pull requests or issues.

Where do you want to take the project? The name service and identity verification side of things on their own seems like they could be very useful (like a P2P Keybase).

I'll probably continue with the messenger as the main product and add simple identity management features. Basically a shared contact list where you can look up people and verify each others' contact details, public keys and all other kinds of attributes. Signing in to websites by scanning a QR code in the app would be a cool and useful feature.

I'll maintain the name service and other core functionality in iris-lib which is easy to integrate with other applications. Would be cool to see integrations with crypto wallets (looking up users' payment addresses) and other apps. I recently created a "live chat" widget that you can embed to your website from iris-lib. Embeddable "comment box without trolls" could be popular with bloggers. You could even use it on a static webpage served over DAT or IPFS.

It seems like Iris is shaping up to be much larger in scope than I originally understood. There's so many potential use cases, it must be a little overwhelming.

Where can people find out more and contribute to the project?

Thank you! It has indeed been a bit overwhelming, and it's taken some effort to focus. I hope group chats and mobile notifications will get things going.

The project's Github repository (`github.com/irislib/iris`) is the best place to find out more and contribute.

Excellent. Best of luck with the project.



HARDWARE HACKER MINI TOOLKIT

As someone who tinkers with custom hardware, I've grown to use a small selection of core tools almost daily. Over time this has morphed into a mini toolkit that can complete various tasks, from soldering, desoldering, and assembling prototypes, to disassembling and repairing electronics. It's inexpensive to put together, and fits inside a small pencil case.

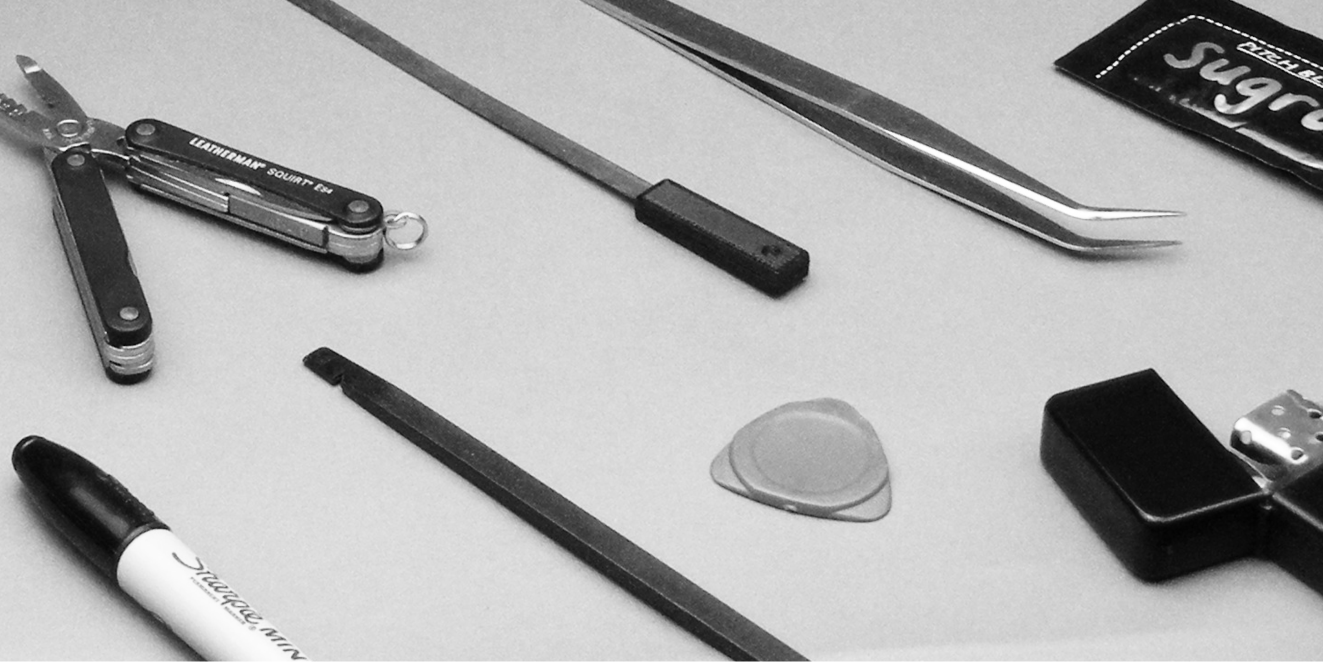
For specialized tasks there are more complex tools and machines available, but this little kit can handle almost all small tasks. It's been indispensable for my work.

USB Soldering Iron. These cheap USB soldering irons are surprisingly good, and are more than capable for light work.

Solder/Desolder Wick. Custom 3D-printed solder and solder wick dispensers allow you to complete small soldering tasks.

Mini Screwdrivers. Medium sized precision screwdrivers in both phillips and flathead are more than suitable for most small tasks.

Sharpie. Standard must-have item.



Leatherman Squirt ES4. Tiny multitool that can cut/strip wires, has files, knife, and pliers. Killer addition to any toolkit.

Lighter. Useful for applying heat shrink, and also sometimes helpful for molding plastics.

Spudger & Pick. Great for getting into electronic casings, prying glued seams, removing goop, etc.

Mini Hacksaw Blade. Literally hack electronics apart. I've added a 3D-printed handle to mine.

Angled Tweezers. Crucial if you want to do a bit of SMD component soldering.

Telescopic Magnet. Handy for collecting stuff that fall into small nooks.

Metal Rule. Not only for measuring, but useful for smoothing and leveling materials.

Sugru. Useful for quick prototypes / repairs.

Heat Shrink. Used for splicing and repairing wires and cables. Throw in different sizes.

DISTRIBUTED VPNS - AN OVERVIEW

The privacy-minded browser company Brave recently announced a new product called VPN⁰, a distributed virtual private network (dVPN). As the name implies, dVPNs behave similarly to traditional VPNs but they don't have a central authority. Users can use the network in both a client and server capacity by accessing network infrastructure to forward their traffic, or acting as an exit node to allow data in and out of the network.

While the distributed VPN concept lends itself well to getting around censorship and geo-blocking restrictions, dVPNs can have different methods of ensuring privacy, reliability, stability, and/or anonymity. dVPNs can offer new concerns to users, such as a case where a bad actor on the network performs malicious or otherwise illegal activity through another network user's gateway. With a traditional VPN, blame falls on the VPN company itself, but this layer of protection doesn't exist when individuals are acting as an edge on the network. Some projects address these issues, while others make no attempt.

Distributed VPNs are not a brand new concept, and many of them are active at various levels of maturity. Below, in no particular order, I've assembled a list of active dVPN projects that are currently operating or in the development stage.

VPN⁰

This forthcoming dVPN from Brave allows exit nodes to whitelist services that they wish to accept traffic from. To avoid eavesdropping, VPN0 relies on zero-knowledge proofs to choose exits by referencing the type of traffic being sent against lists of potential nodes that permit it. The exit doesn't know the specific type of traffic it is carrying, while the client doesn't know the full list of services the exit allows. VPN⁰ has a primary goal to preserve privacy, but also places importance on network performance and speed.

`https://brave.com`

ORCHID

Orchid is a dVPN that allows users to buy bandwidth from a global pool of users operating as service providers. Using their own VPN protocol built on top of WebRTC, Orchid allows for clients to buy bandwidth while node operators can sell it. OXT, Orchid's own digital currency, is used to facilitate transactions while the Ethereum blockchain is leveraged for providers to advertise their service.

`https://www.orchid.com`

MYSTERIUM

Mysterium is an open-source dVPN built on Ethereum that can integrate with OpenVPN and WireGuard. While the network is currently free to use, users can choose to run nodes operating as traffic relays and receive ETH automatically from Mysterium based on node performance. To protect operators, a whitelist is in place to restrict traffic that goes out of the network and block malicious activity.

`https://mysterium.network`

LOKINET

The Lokinet is a decentralized VPN that utilizes onion routing to provided encrypted routing within and outside of the network. Network operators automatically earn $Loki, a cryptocurrency based off of Monero, by operating service nodes on the network. While service nodes can simply provide internal resources, they receive more compensation for allowing exit traffic to pass through them.

`https://loki.network`

SENTINEL

Sentinel is a decentralized VPN backed by blockchain technology to incentivize users on the network to run router nodes. Using a "Proof of Traffic" consensus system, potential bandwidth is advertised and tracked for incentive payout. Currently, the Sentinel dVPN is in a proof-of-concept phase and aims to be a solution for safely routing user's traffic.

`https://sentinel.co`

HOLA

Perhaps the longest-operating dVPN (formed in 2007), Hola creates a network where each user is both a client and an exit.

Hola additionally implements peer-to-peer caching, where popular content may be cached within the network to reduce the need to perform multiple external retrievals and increase network speed.

More recently, users have to option to enroll in a paid tier of service wherein they can use the network without acting as an exit.

`https://hola.org`

PRIVATIX

Built on Ethereum, Privatix is a dVPN that allows purchase/sale of bandwidth between consumers (clients) and operators (agents).

Using smart contracts, zero-trust payments can be made to node operators with the PRIX currency, all without revealing the IP addresses of either party. Clients and agents are connected to one another using onion-routing, which prevents actors on the network from learning who is on either side of any given connection.

`https://privatix.io`

LETHEAN

Lethean is a dVPN where clients seek out exit nodes they want to use and purchase bandwidth from them with the Lethean cryptocurrency, based on CryptoNote.

Using a marketplace, exit node operators can advertise their country, speed, limitations, etc. so clients can choose an exit that will best work for them.

While both clients and nodes can access the VPN via Lethean wallet applications, clients can also utilize the VPN through browser plugins for Firefox, Chrome, Opera, and Brave.

`https://lethean.io`

NYM

Nym is a dVPN utilizing mixnet technology to provide anonymity and security over the network. After a packet is encrypted, the Sphinx packet format is used to pad all packets to the same size and make them indistinguishable from one another.

Nodes within the network will delay the transmission of these packets and introduce dummy packets that create a constant stream of traffic to disrupt observers. Node operators are rewarded automatically in a proof-of-stake system for how much traffic they mix as part of the network.

`https://nymtech.net`

VPN GATE

VPN Gate is an academic project that creates a dVPN to combat censorship and surveillance imposed by governments.

Currently, VPN Gate has over 5000 volunteer run nodes, and users can connect to the network through a variety of protocols including OpenVPN and L2TP/IPsec.

Exit node operators must operate on residential connections with dynamic IP addresses, as governments are known to block ranges of IP addresses owned by datacenters and hosting companies.

Further, VPN Gate is offered free of charge, and does not require sign-up.

`https://vpngate.net`

FAREWALL TO PIRATEBOX AND A PROPOSAL FOR THE FUTURE

One of the earliest NODE projects was creating a 'Pocket Piratebox', basically modifying the standard TP-Link MR3020 Piratebox install by removing its case and wiring it up to a battery so it fits in your pocket. This was super simple, and a good beginner hardware hacking project.

The original Piratebox project always held a special place in my heart—the idea of creating mini WiFi access points in order to share files or set up adhoc, in-person chatrooms seemed really cool, and I was saddened to find out that not only has development stopped, but that the `Piratebox.cc` site will be going offline sometime in 2020.

According to lead dev Matthias Strubel, this shutdown is caused by a range of factors, mostly revolving around lack of time and lack of external help. Regardless of the reasoning, I have big respect for Matthias as he's been diligently working at this for many years, and I look forward to seeing what he does next.

A PROPOSAL: COMMUNITYBOX

Something I've been pondering for a few years is creating a Piratebox spinoff with a slightly different take, and since Piratebox is now going away, I thought maybe it's time for something new to step in. I'm tentatively calling it CommunityBox.

The idea is to create offline social networks aimed at local in-person communities, allowing users to chat, share news, and organize without needing ISPs or services like Facebook and Google.

You could get a few people in an area to run CommunityBoxes, making sure their WiFi signals overlap. Each box would automatically connect to others in a peer-to-peer mesh configuration, so say when a new calendar item is uploaded to one, it's shared with the rest of the nodes. This allows them to not only

cover smaller areas like apartment buildings, hackerspaces streets, and neighbourhood watch groups, etc. but also to expand to larger areas such as conferences, multiple city blocks, and fluid moving locations like protests.

Users within WiFi range could connect to the open access points using their phones and computers in order to keep up with what's going on and to contribute.

Think of it like a digital community noticeboard where you have to be physically in the community in order to access it.

HUMAN ORIENTED

One of the interesting side effects of our hyperconnected world is the increased atomization and loneliness many of us feel.

The famous "Dunbar's Number" posits that the size of human brains has a cognitive limit with regards to the number of stable social relationships one can have—that number being 150—and anything above that adds extra complexity and problems.

Throughout all other times in human history, we've had to rely on our immediate communities around us in order to survive, yet in this digital age, we have gone so far in the other direction that many of us don't even know our neighbours anymore. Paradoxically, lots of people have hundreds or thousands of 'friends' on social media.

What if something like a CommunityBox could help re-establish some of that in-person community spirit, even in a small way?

HARDWARE IDEAS

In order to develop a project like this, we'd need to concentrate on using hardware that strikes a good balance between availability, price, and features.

As you have probably guessed, I think Raspberry Pis are the front-runner for this kind of thing since they work well as a low power wireless server.

Continued →



At its most basic installation, a non-technical user should be able to write the micro SD card, insert it, plug everything in, and boot up.

You can also attach long-range WiFi antennas, and WiFi interfaces to increase the coverage. Aside from that, Raspberry Pis can easily interface with LoRa modules for longer range communications, so that might be an area to explore further?

The low power consumption of a Raspberry Pi means it's possible to have backup battery setups, as well as full off-the-grid solar/wind boxes. This would be particularly handy in emergency scenarios, or just when power or the Internet goes down, allowing communities to continue communicating.

GET INVOLVED

So what do you think? Does this sound like a project you'd want to contribute to? I'm not sure what specific peer-to-peer architecture we'd use for this, but we can figure it out. If that’s your area of expertise, get in touch.

Likewise, if you're a good web designer, we'll need help designing the user interfaces. It'd be sweet if we could also design custom hardware add-ons for battery/solar options and rugged enclosures. So, if that's in your wheelhouse, I'd like to hear from you.

I have registered the `communitybox.org` domain for when things start moving, but in the meantime, if you are interested in helping or testing, email me at `mail@n-o-d-e.net` and we can try and put a small team together.

Hopefully we can create something useful for communities around the world, and keep the sentiment alive that the Piratebox project fostered for all those years.

THE AXIOM OPEN HARDWARE CAMERA

Since 2012, filmmaking and camera technology enthusiast Sebastian Pichelhofer has been working on the AXIOM open hardware digital camera under the Apertus non-profit organization. Given the short timeframe and resources available, what has been developed so far is remarkable.

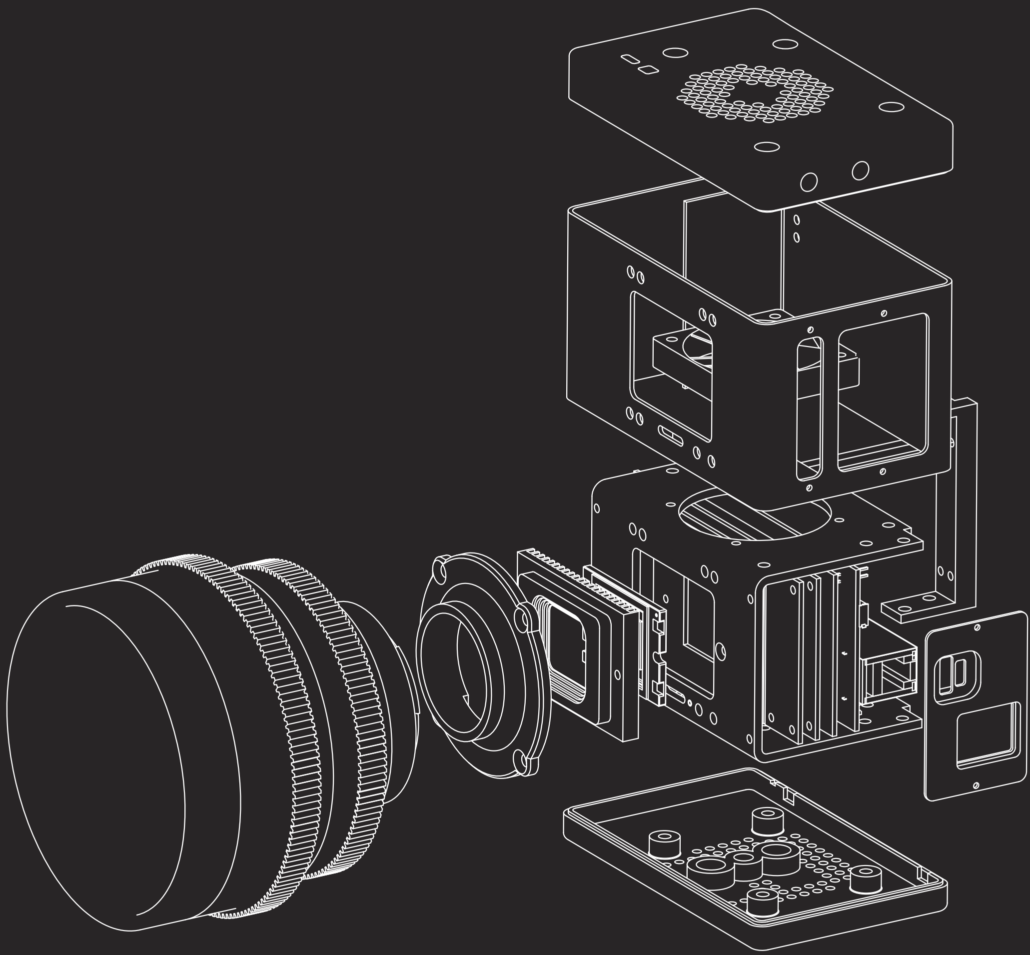
When most people think of digital cameras, they often think of their smartphone or a DSLR, but Sebastian wanted to create a high-end camera for cinema. This isn't influenced only by his love of film, but due to economic reasons as well.

"The AXIOM Beta can be used for capturing still images but considering the price-tag of our image sensor alone being way beyond 1000€ per piece, it's difficult for a small team like us to create anything that competes in terms of quality and price with what DSLRs from Canon, Sony, Nikon, Panasonic, etc. are producing already. When it comes to moving images the situation is completely different though," Sebastian explains.

This decision to go with video over still images makes sense, and has definitely drawn in other like-minded people who agree, with the project expanding from a few contributors to a large, worldwide team building the AXIOM.

The sensor in the Beta unit is capable of capturing moving images at up to 300 FPS in full 4K resolution, but that's just scratching the surface of the appeal of this camera. Since the project is both open hardware and software, the level of configurability and modularity is huge. It basically gives you a top quality base platform that you can tweak and customize for your specific needs.

The internals of the AXIOM consist of a bunch of different PCBs, designed for plugging and swapping parts. This means you can choose the type of video output ports, USB 3.0 output, SSD adapters, ethernet ports, and more. You can also plug in regular USB devices such as thumb drives and wifi adapters, and they will work natively on its Linux-based OS.




AXIOM
1/4
MOUNTS



AXIOM
1/4
FUTURE EXTENSIONS
JTAG
FIRMWARE
USB UART
STORAGE
USB
ETHERNET
1/4
PLUGIN MODULE 1
PLUGIN MODULE 2
MOUNTS
AXIOM
BETA





The software side of things is completely open too, giving users many ways to interface with the device. Protocols such as FTP, and SCP are all available, as well as an in-development HTTP GUI, allowing access to the camera from phones, tablets, or computers.

Apertus' communications lead RexOr sums it up nicely, "Everything has been done on virtually no money and those who're experienced with electronics endeavours of this scale view what's been achieved thus far as a minor miracle." I definitely agree, the level of output here is phenomenal and in my mind the project is a testament to the power of collaboration and open hardware/software.

At the time of writing, the AXIOM Beta is still in the testing phase, with about 50 units being tested worldwide. The team is aiming for production and assembly Q1 of 2020, with shipping in the second half of the year.

If you want to learn more, and maybe get involved, check out Apertus' site: `https://apertus.org`

If you're a reader of this zine, you should have a grasp of why peer to peer software has such great potential. The ability for us to regain control of our digital lives with networks and platforms that the users themselves own and operate is a big deal, so the question is, why aren't P2P systems more widely used? I asked some

leaders in P2P tech their thoughts, and this is what they said.

"They need to have better UX and better features than the centralised alternatives. It's way too easy to drown in the details and complexities of decentralisation at the expense of UX. Riot has suffered from this a lot, but we are now determined to fix it at any cost. On the plus side, a well-designed open decentralised ecosystem (e.g. the web) should always be able to beat a silo on features via the size and vibrancy of its community."

Matthew Hodgson
Matrix Co-Founder

"For decentralization technologies to achieve broad adoption, it's critical that they deliver an experience that is as good as—or better than—the solutions they aim to replace. Traditional users generally won't care about decentralization if it comes at the expense of performance, cost, reliability, user experience, or the myriad of other considerations in the buying process. Decentralized solutions need to surprise and delight the user. When they can achieve lower cost, higher performance, and better reliability, then you will see P2P technologies take off."

Shawn Wilkinson
Storj Founder

"I was told I have 500 characters to explain what needs to happen to P2P apps, but I only need one [word]: usability.

The ceiling on the number of people who will use apps for ethical, technical, or privacy reasons is a [small] percentage of the population (I say this as a member).

The ceiling on the number of people who will use apps because they deliver a better experience is only limited by the rate of human reproduction."

Jeremy Kauffman
LBRY Founder

"Better user experience. Barriers to entry continue to fall as devices get cheaper and better, but there's still a knowledge barrier keeping out non-technical users.

Benefits beyond ideology. Current P2P users are mostly "true believers." The more that people are able to experience tangible benefits, the more adoption we'll see. Cheaper, faster, more private, more secure, censorship-resistant—any benefits at all—need to be emphasized more than our personal beliefs."

Sam Patterson
Open Bazaar Co-Founder

“We have to stop trying to sell P2P as a feature—it's an implementation detail—and focus on building legitimately good products that happen to be P2P.

Yes, the discerning, educated user cares about P2P, but most users are casual, thus don't, and this isn't a bad thing!

We're already playing at a disadvantage considering P2P is harder to build. If we don't acknowledge this and act on it, it becomes infinitely harder to even be in the fight for the average user's attention compared to centralised alternatives.”

Burak Nehbit
Aether Founder

“I think the content is the key to make P2P technologies be more widely used. As for developers: We have to make it as easy to use and as fast as the centralized Internet for the users and for the developers as well. As for community: one thing that that we could do is create sites that describes each project, lists the available content and the possible use cases.”

Tamas Kocsis
Zeronet Founder

THE REFORM 2

Last issue we showed you the Reform, an awesome open hardware laptop project by Lukas F. Hartmann. In the intervening year, he's taken all the feedback and lessons learned from V1, and is back now with an almost completely redesigned Reform 2.

UPDATES

The i.MX6 CPU has been upgraded with the faster i.MX8, and the system on chip previously used is now replaced with a fuller featured system on module board which includes RAM, USB, PCIe interfaces, Ethernet chip, etc. directly on the board.

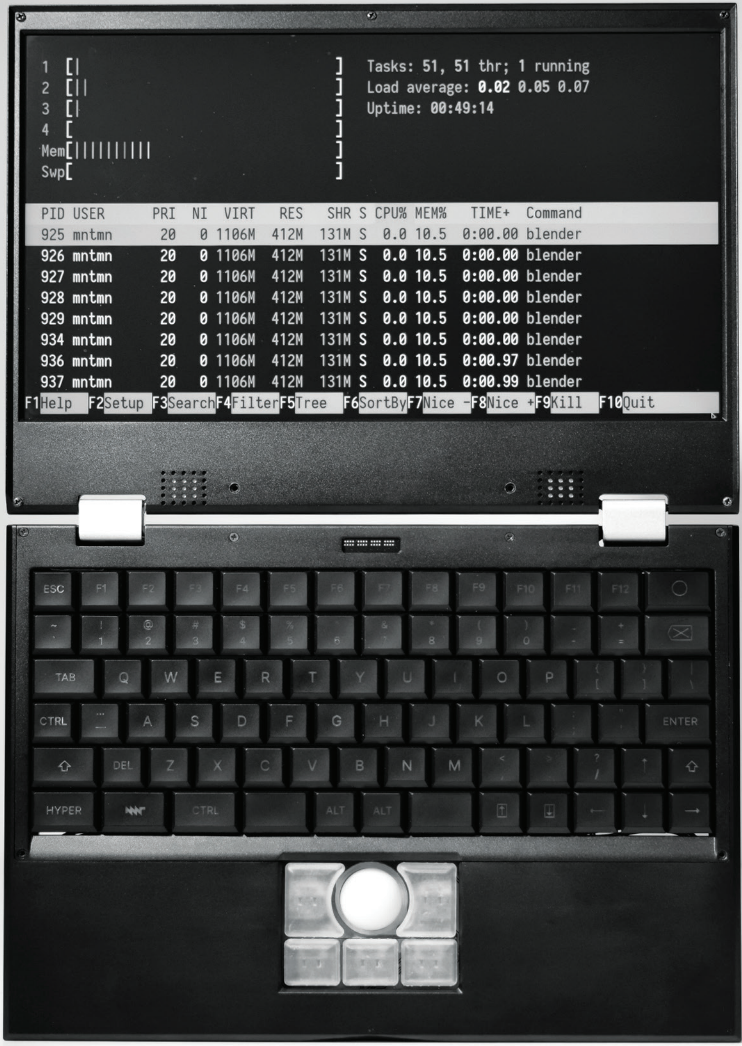
Lukas explains, "We selected their 'Nitrogen8M SOM' module for Reform because it is the only available module for which you can download the complete schematics and understand what every component does. This means that anyone will be able to design a replacement SoM to power Reform with a completely different CPU or an FPGA, for example." All this is secured within the newly-designed motherboard and slots in directly via a 200 pin SO-DIMM connector.

The power system used has also been upgraded, with Lukas teaming up with Fully Automated Technologies, another open hardware project, to deliver a pretty beefy 28.8V operating voltage via 8x 18650 batteries connected in series.

One of the cool things about this open design is that if any of the cells fail in the future, you need only replace them at ~€2.50 per cell, instead of replacing the entire thing (looking at you Apple).

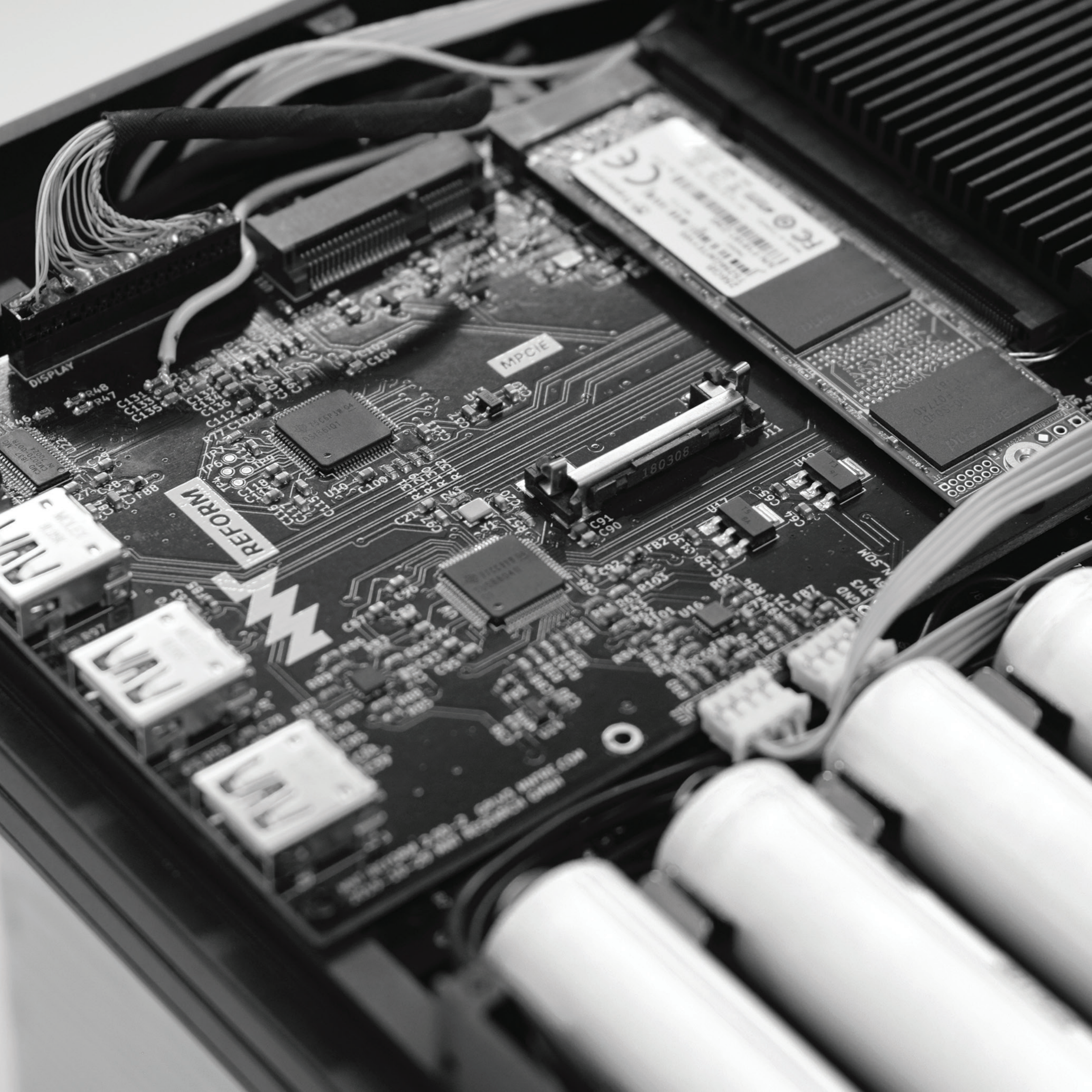
In terms of ports and expansion capabilities, the Reform 2 has gigabit Ethernet, 3.5mm headphone/mic jack, a bootable SD card port, HDMI port, and 3x USB 3.0 Type A high speed ports, while internally there's a m.2 slot for an SSD, and an mPCIe expansion slot for WiFi cards, graphics cards, cell modems, etc.

Instead of being fully 3D-printed, this time the keyboard features blank Kailh keycaps that


1 [|
2 [||
3 [|
4 [
Mem[||||||||||
Swp[
Tasks: 51, 51 thr; 1 running
Load average: 0.02 0.05 0.07
Uptime: 00:49:14
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
925 mntmn 20 0 1106M 412M 131M S 0.0 10.5 0:00.00 blender
926 mntmn 20 0 1106M 412M 131M S 0.0 10.5 0:00.00 blender
927 mntmn 20 0 1106M 412M 131M S 0.0 10.5 0:00.00 blender
928 mntmn 20 0 1106M 412M 131M S 0.0 10.5 0:00.00 blender
929 mntmn 20 0 1106M 412M 131M S 0.0 10.5 0:00.00 blender
934 mntmn 20 0 1106M 412M 131M S 0.0 10.5 0:00.00 blender
936 mntmn 20 0 1106M 412M 131M S 0.0 10.5 0:00.97 blender
937 mntmn 20 0 1106M 412M 131M S 0.0 10.5 0:00.99 blender
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit


V
ALT
ALT


REFORM
F6
F7


DISPLAY
MPCIE
REFORM
180308



can either remain unlabeled, be laser engraved, or UV inkjet printed.

The Reform 2 display has been upgraded to a larger 12.5inch, 1080p panel by Innolux, and there's also a handy little 128x32px OLED on the keyboard which can check the charger and battery status without needing the operating system. Lukas explains, "The keyboard OLED and direct interaction mechanism has more potential future uses, like a password manager, crypto wallet or notification display".

And finally, the entire thing is contained within a newly-designed case by industrial designer Ana Dantas. The black anodized and sand blasted aluminum enclosure is slick, and provides some extra robustness over the previous 3D printed case.

Like the previous design, this version's CAD, PCB, 3D-print files, firmware, schematics, and bill of materials are all available at:

`https://source.mntmn.com/MNT/reform`
`https://mntre.com/reform`

SECURE SCUTTLEBUTT - DECENTRALIZED SOCIAL MEDIA & MESSAGING

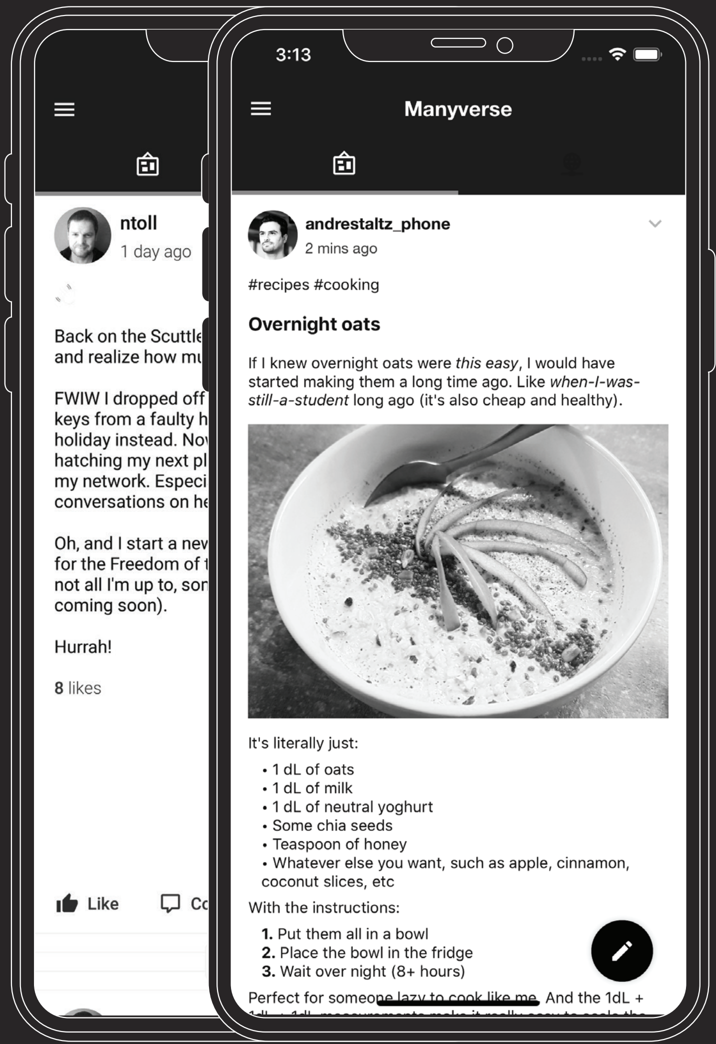
Secure Scuttlebutt (`scuttlebutt.nz`) is an open-source protocol used for messaging and content sharing, with no reliance on centralized servers. Starting in 2014, Secure Scuttlebutt (SSB) aims to be a platform where users can self-host content and exchange it with peers in a secure, fault-tolerant manner.

HOW IT WORKS

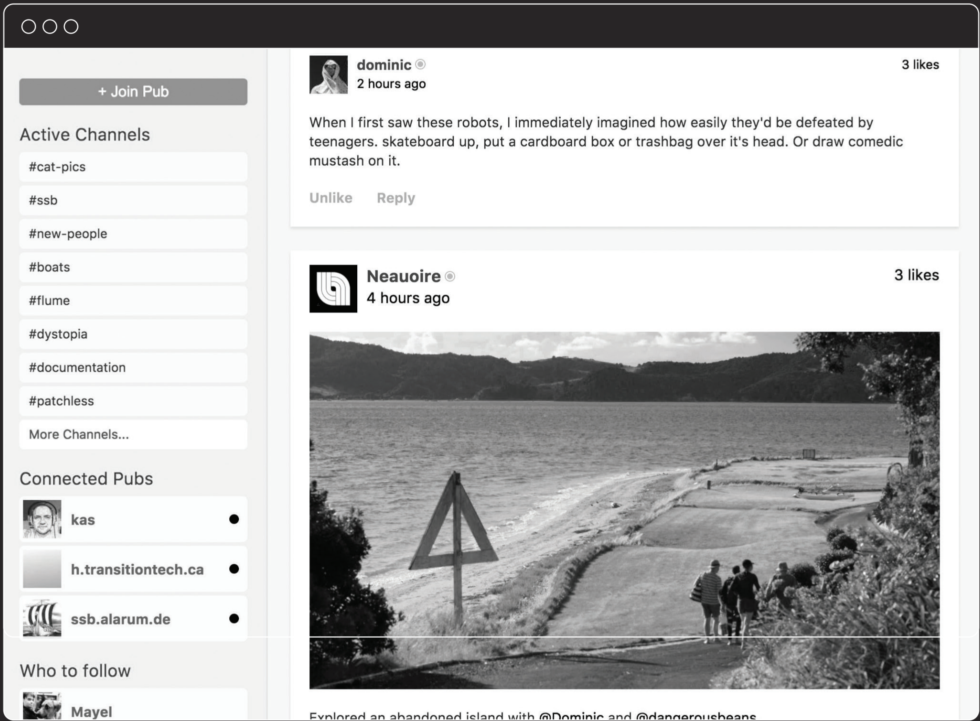
The heart of Secure Scuttlebutt is a database comprised of entries from message feeds. Much like with Bitcoin and other cryptocurrencies, SSB feeds are immutable, append-only blockchains. Nobody can go back and modify previous messages, and messages cannot be "forgotten." Only the owner of any given feed has permissions to write to that feed, which is enforced by digital signatures.

SSB operates on the principle of "offline-first," meaning that data is established and kept on the machine that originates it. After the data is written locally, it is then made available for others. With each user operating at least one node within the network, Secure Scuttlebutt tends to mirror real-life person-to-person interactions within a population, via a P2P network topology. Consider the use case of a group of people who want to exchange messages with one another. Each person would run an SSB-based messaging application on their own machine that will write their outgoing messages to a feed. Then, each person would follow the feeds of everyone else they wish to message.

Data is exchanged via SSB through the use of a peer-to-peer global gossip network. This functions similarly to how a rumor might spread through an office. When one peer wants to get updates for a particular feed from another peer, the requesting peer is given more messages for other feeds within



ntoll
1 day ago
Back on the Scuttle
and realize how mu
FWIW I dropped off
keys from a faulty h
holiday instead. Now
hatching my next pl
my network. Especi
conversations on he
Oh, and I start a new
for the Freedom of t
not all I'm up to, sor
coming soon).
Hurrah!
8 likes
Like
3:13
Manyverse
andrestaltz_phone
2 mins ago
#recipes #cooking
Overnight oats
If I knew overnight oats were this easy, I would have started making them a long time ago. Like when-I-was-still-a-student long ago (it's also cheap and healthy).
It's literally just:
• 1 dL of oats
• 1 dL of milk
• 1 dL of neutral yoghurt
• Some chia seeds
• Teaspoon of honey
• Whatever else you want, such as apple, cinnamon, coconut slices, etc
With the instructions:
1. Put them all in a bowl
2. Place the bowl in the fridge
3. Wait over night (8+ hours)
Perfect for someone lazy to cook like me. And the 1dL +


+ Join Pub
Active Channels
#cat-pics
#ssb
#new-people
#boats
#flume
#dystopia
#documentation
#patchless
More Channels...
Connected Pubs
kas
h.transitiontech.ca
ssb.alarum.de
Who to follow
Mayel
dominic
2 hours ago
3 likes
When I first saw these robots, I immediately imagined how easily they'd be defeated by teenagers. skateboard up, put a cardboard box or trashbag over it's head. Or draw comedic mustash on it.
Unlike
Reply
Neauoire
4 hours ago
3 likes

their social circle that they may or may not be aware of. This contributes to a characteristic of the protocol known as eventual-consistency. Over time, peers within a social circle (direct friends, and sometimes friends of friends) will have all messages shared with one another. With the network architecture being set up like this, peers do not need to have direct connections between one another to exchange content, and can instead use intermediaries to forward data to an intended peer.

USAGE

While SSB itself has been adapted to many uses such as music sharing, a git subsystem, a chess application, and more, its most popular use is for social networking.

Applications running on Secure Scuttlebutt can be thought of as different views into the greater Scuttleverse, the decentralized network of SSB peers.

There are many different social networking applications that use Secure Scuttlebutt. The most popular is Patchwork, a desktop application built on Node.js. Similar applications like Patchfox, a social networking client as a Firefox WebExtension, and Manyverse, a social networking client for mobile devices, will allow for similar views into the Scuttleverse.

Social networking applications on the Scuttleverse look and function much like traditional centralized social networks such as Facebook or Twitter. Users create an account, follow other users, private message (securely, using public-key cryptography), and send/receive posts, photos, and likes while participating in various conversation threads. All data is first written to and stored on the user's device, and then exchanged to other peers over the Internet or on a local network.

Manyverse in particular is an intriguing application for the Scuttleverse. By running on mobile devices, Manyverse allows for a new method of social communication between people in Africa, Asia, and Latin America who may have cheap mobile phones but limited access to the Internet. While Manyverse allows for data exchange over the Internet, it can also use a WiFi or Bluetooth connection to



Secure Scuttlebutt Consortium Member & Manyverse Creator, Andre Staltz

facilitate communication. This makes it ideal for off-grid living, communication in the event of a natural disaster, or for use when the Internet is censored or disrupted.

ADVANTAGES

Due to the offline-first concept and gossiping nature of the protocol, SSB is ideal for those with unreliable access to the Internet, such as sailors, nomads, or people living off the grid.

As previously mentioned, SSB makes identity verification easy, as messages can only be appended to feeds with the digital signature of the author. Forging messages is not possible.

Additionally, it aids in the prevention of spam. Users will only download messages from peers they follow (or optionally peers of peers), minimizing unwanted content.

DISADVANTAGES

Due to the fact that a peer on the network is ingesting feeds and creating a local cache for offline access, Secure Scuttlebutt can take up a lot of disk space. Additionally, it can take a long time for messages to propagate throughout the network. SSB could be used for real-time communication, but it more suited to non-urgent messaging.

Further, due to the decentralized nature of SSB, new participants in the Scuttleverse might be isolated from other peers because there aren't any users nearby. To mitigate this, SSB has the concept of "pubs," which are public peers on the network that follow any user back when they themselves are followed. This allows for bootstrapping an initial community, but is an additional step in getting on the network.

CONCLUSION

Secure Scuttlebutt is a new take on how people can communicate online, without the need for centralization or always-on Internet access. SSB, and the applications that use it, become the must-have technology for anyone spending time off-grid who still wants to share information in a secure, reliable way.

MAKING MEANING

Do you have a 3D printer sitting around collecting dust? Want to make something more than toys and other useless trinkets? Perhaps you have skills in design, programming, or engineering that are being wasted on unfulfilling bullshit?

There is a space out there waiting for you to fill it: you can use your knowledge to work on important things that will help others.

Some of you may already be in that place, but for most of us who are searching, here are some open source projects with big goals. Maybe one of them will catch your eye or spark an idea of your own?

Many of them are tangibly contributing to improve the world, and all could do with more help.





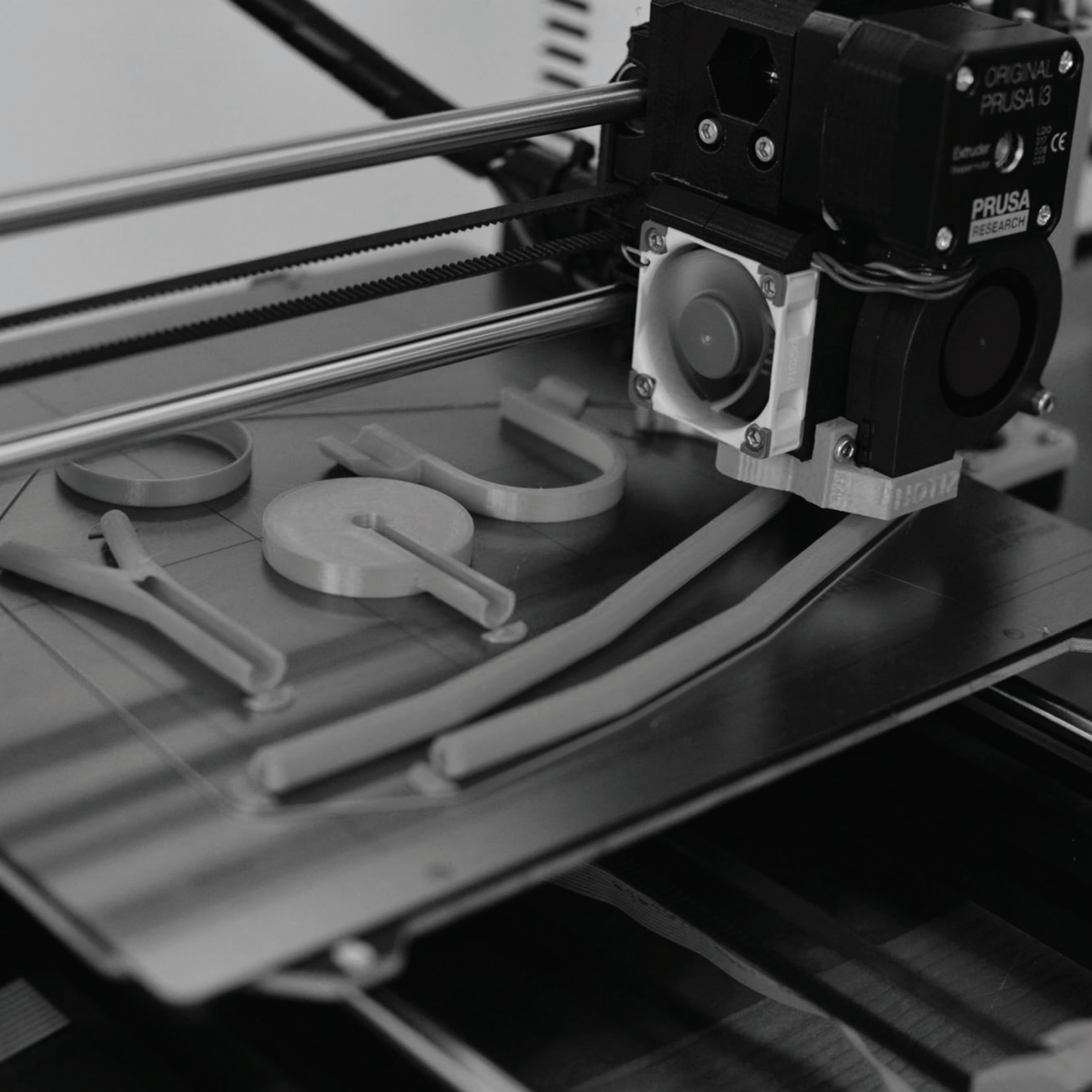
THE GLIA PROJECT

The Glia Project's goal is to create a range of high-quality, low-cost open source medical devices that can be manufactured in low-resource settings.

So far the team has created a variety of 3D printable tools including stethoscopes, otoscopes, and tourniquets. Each item has been tested in the field, all around the world with Glia's own doctors and medical students.

The team are also currently working on a pulse oximeter for measuring oxygen saturation, an electrocardiogram for measuring the electrical activity of the heart, and kidney dialysis device.

`https://github.com/GliaX`
`https://glia.org`


ORIGINAL PRUSA i3
Extruder
PRUSA RESEARCH



E-NABLING THE FUTURE

Started in 2011, the e-NABLE project matches those with full and partial upper limb amputations and disabilities with 3D printer owners and designers to provide custom fitting, functional prosthetics.

There is a large community of around 20,000 volunteers in over 100 countries that have provided an estimated 8,000 custom prosthetics to both children and adults, giving them some degree of autonomy back to their lives.

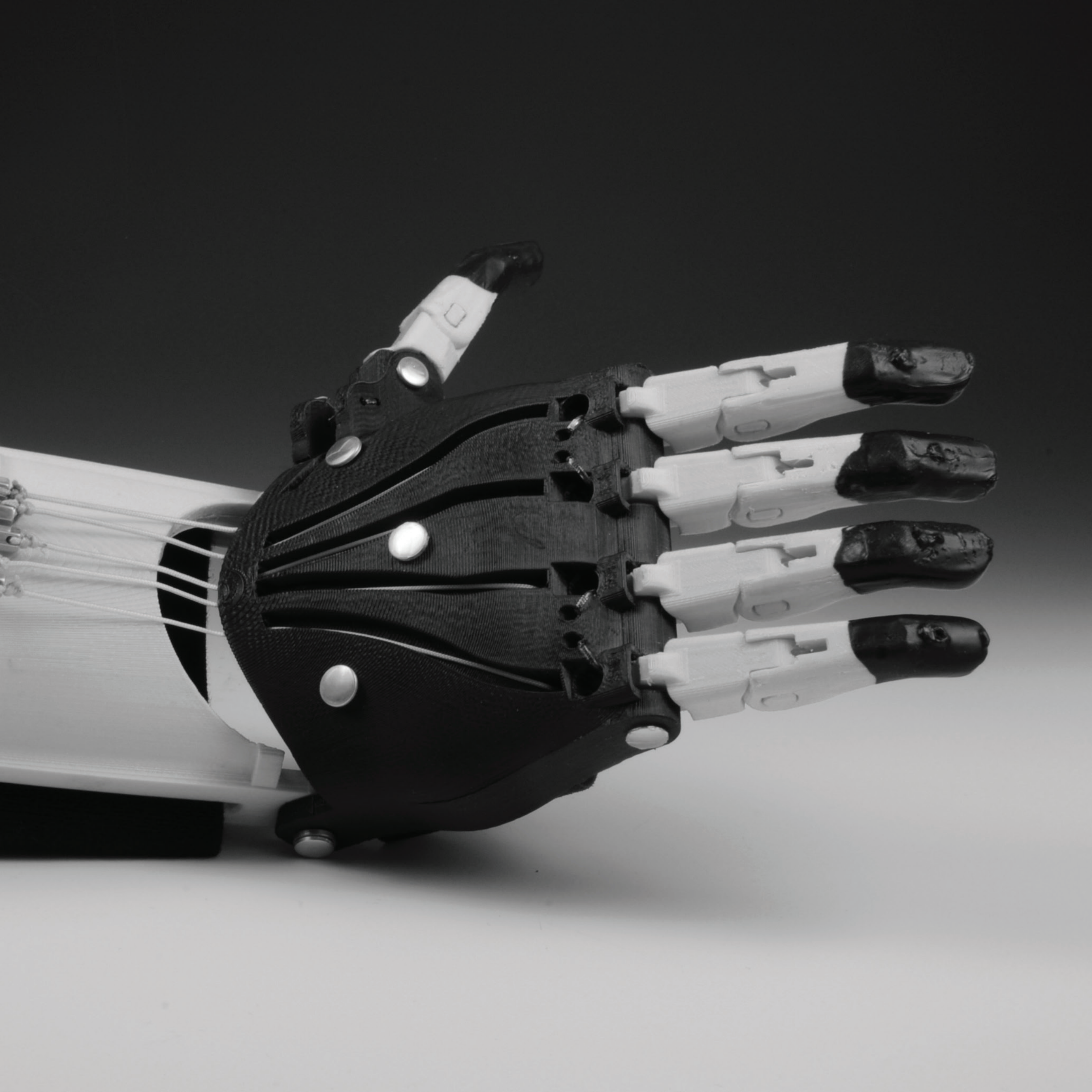
This is particularly helpful for children as it can get expensive for parents to keep replacing them as the kids are constantly outgrowing their existing prosthetics.

The project site has a detailed map showing local chapters all over the world, so it's really easy to find collaborators and people in need who live nearby to you.

`https://enablingthefuture.org`









ALICE EXOSKELETON

Created by the Mexico-based design and engineering group Indi, the ALICE project is an open source lower leg exoskeleton aimed primarily at assisting the mobility of children.

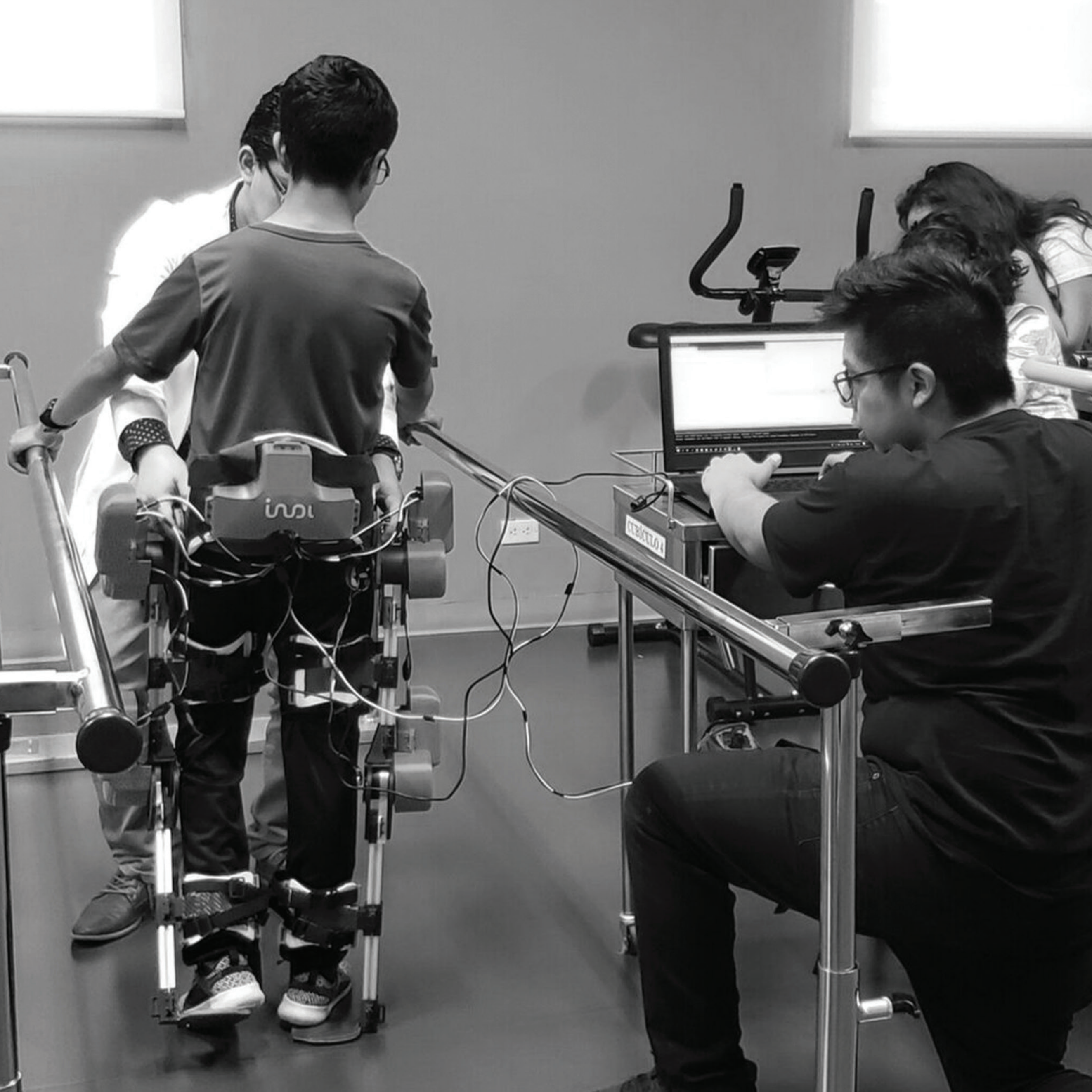
After six years of research and development and over $60,000 invested in the project, the team have released a fully working exoskeleton design that's available to the world under the CC BY-NC license.

The open source designs for the ALICE can be requested through the Indi website, with priority given to engineers and clinicians, due to the potential danger if used incorrectly.

The bill of materials is around €1500, making it pretty accessible for a high tech piece of kit.

This current design is in clinical use.

`https://www.indi.global/alice`



OPEN SOURCE BIONIC LEG

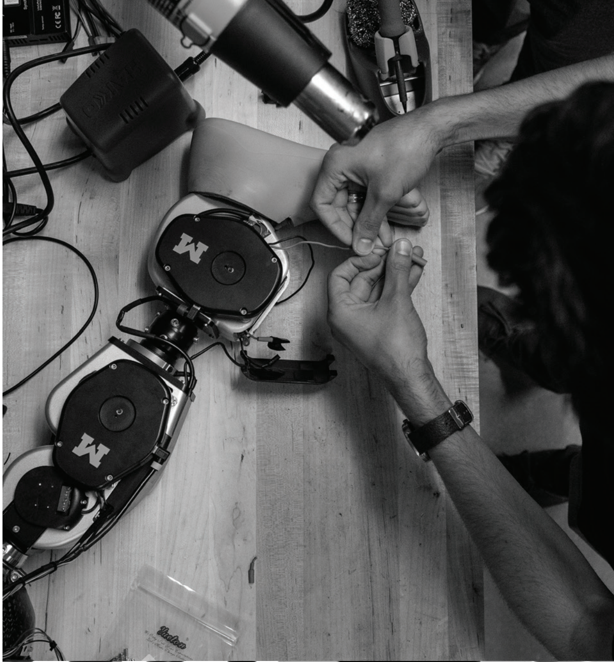
The Open Source Bionic Leg is a project collaboration between researchers at the University of Michigan and the Shirley Ryan AbilityLab research hospital.

The team has come up with a fully articulating robotic leg, complete with automatic knee and ankle movement and the ability to seamlessly switch between activities like walking, going up stairs, or down ramps. This function is all thanks to the Raspberry Pi which runs inside.

The current bill of materials is around $28k, which includes custom machined parts.

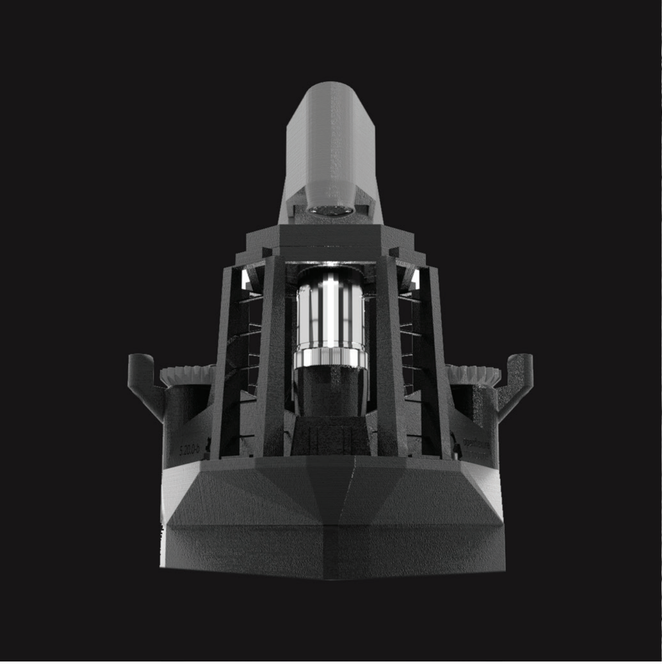
All the files and instructions are available on the project site.

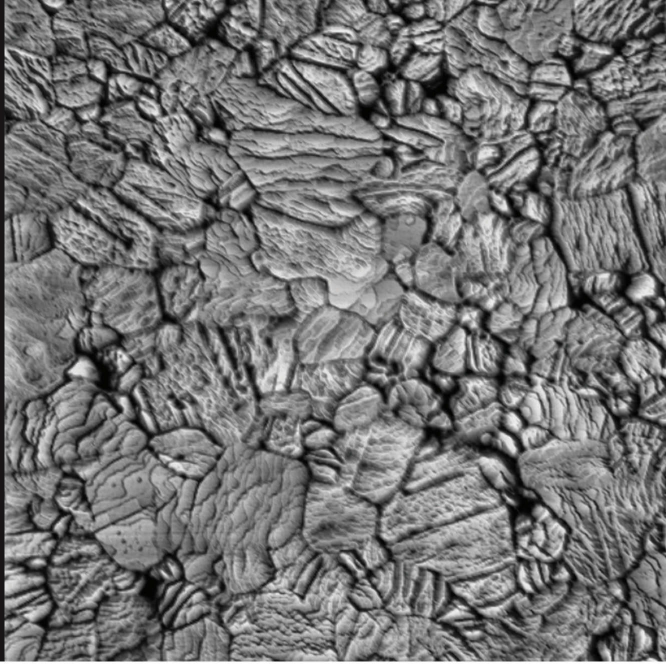
`https://opensourceleg.com`











OPENFLEXURE

The OpenFlexure Project makes high precision mechanical positioning available to anyone with a 3D printer.

The first project the team released was the OpenFlexure Microscope, a little 3D-printable, Raspberry Pi-based optical microscope, perfect for scientific experiments.

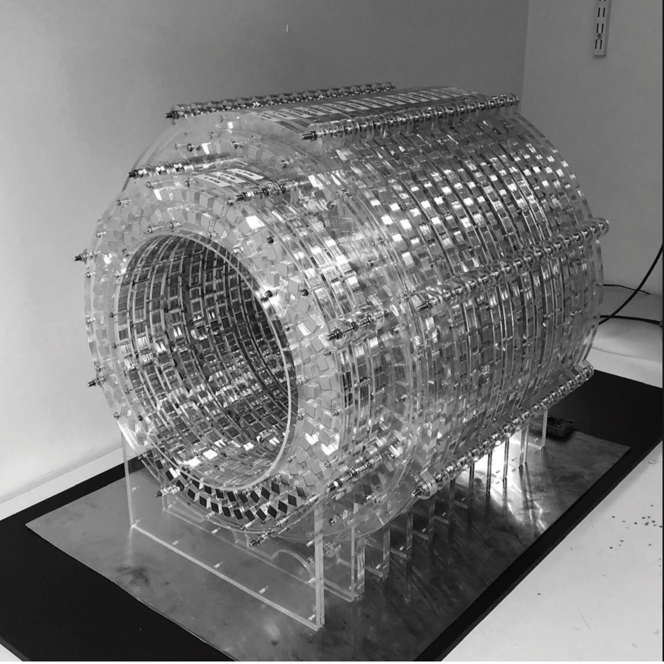
`https://openflexure.org`

ATOMIC MICROSCOPE

This home-built scanning tunneling microscope project by Dan Berard has developed a fully working microscope, capable of atomic resolution imaging in air.

Full details and source code for the project are on Dan's website, so check it out.

`https://dberard.com/home-built-stm`



OPEN SOURCE IMAGING

The Open Source Imaging team have been doing important work in the areas of magnetic resonance imaging, aka MRI.

They have a range of projects currently running including both advanced imaging software and fully open source MRI machines.

`https://opensourceimaging.org`



OPENPCR

PCR, or the polymerase chain reaction, is a method of copying DNA molecules, and the OpenPCR is a $500 open hardware machine designed just for that task.

Often used for detecting infections & diseases, testing water/food safety, PCR can also be used to manipulate DNA and more.

`https://openpcr.org`


GILDAN





PRECIOUS PLASTIC

Precious Plastic's goal is to look at technological solutions to plastic pollution around the world.

Started in 2013 by Dave Hakkens, the project has grown to a community of hundreds and has spawned multiple machines that can recycle and repurpose plastic in meaningful ways.

These include shredders for turning plastic waste into tiny flakes, extruders that convert the flakes into long lines of plastic material ready for injection moulding or pressing into new products.

The bustling worldwide community behind the project has also created an ever-expanding range of machines and products based off all of this.

Well worth checking out.

`https://preciousplastic.com`

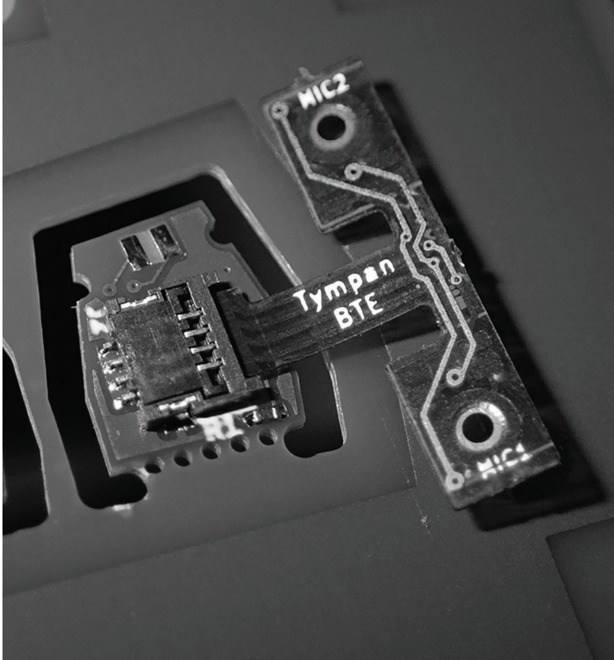
TYMPAN

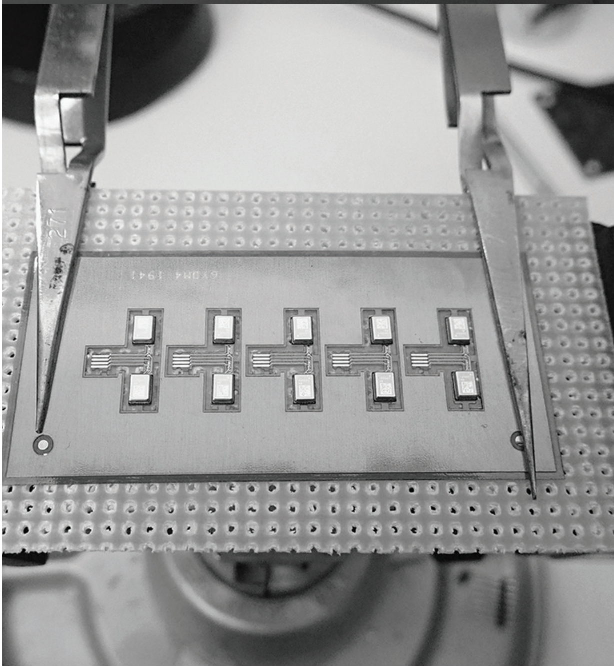
Tympan is an open source hearing aid development platform built on top of Arduino IDE compatible hardware.

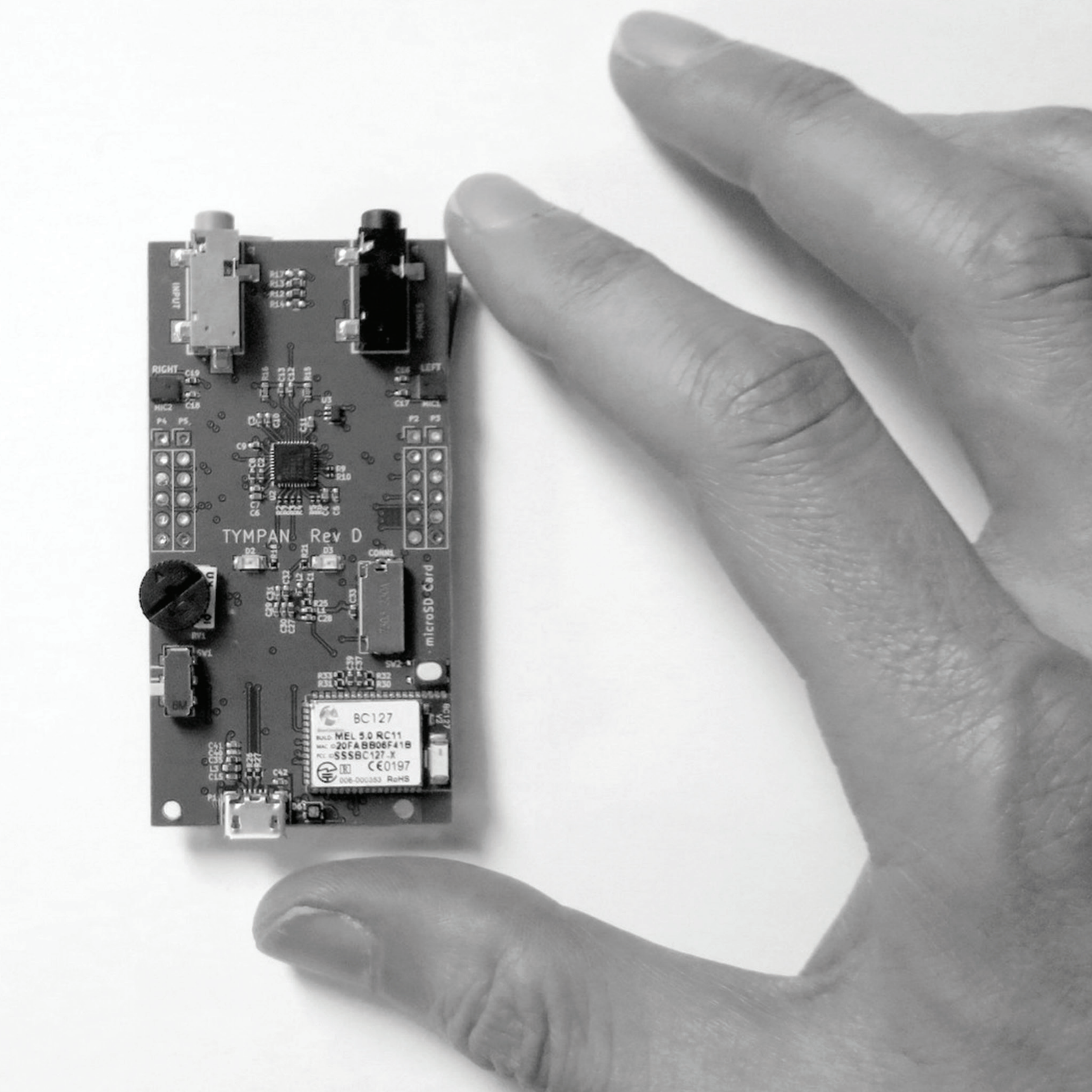
The goal of the project is to lower the barrier for software and hardware refinement, translating those advancements into usable products such as hearing aids, cochlear implants and other consumer electronics.

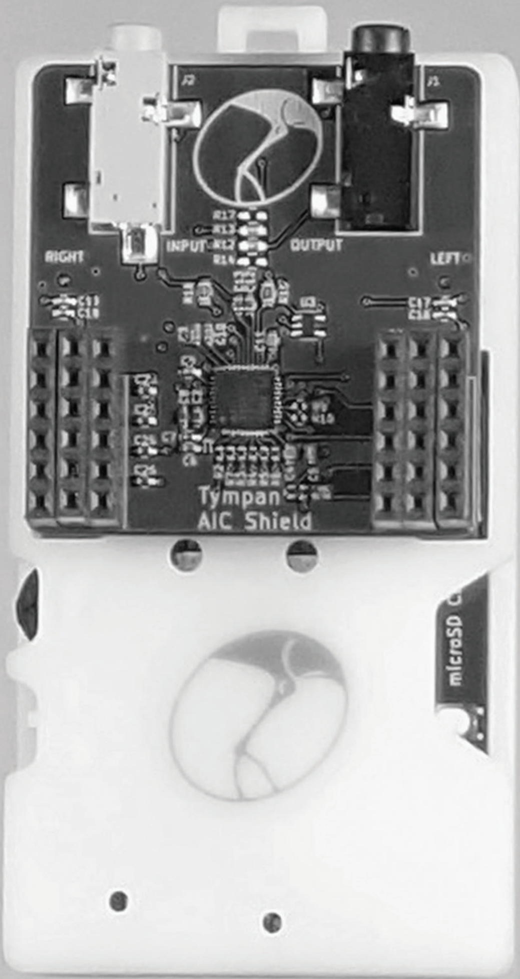
Currently the Tympan is on revision D, boasting a Teensy 3.6 processor, Bluetooth interfacing, 2 on board Knowles MEMS microphones, and expansion options for other microphones or speakers/headsets.

`https://tympan.org`






INPUT
RIGHT
LEFT
TYMPAN Rev D
microSD Card
BC127
MEL 5.0 RC11
20FABB06F41B
SSSBC127-X
CE0197
RoHS


RIGHT
INPUT
OUTPUT
LEFT
Tympan
AIC Shield
microSD

OSBEEHIVES

Open source beehives and colony monitoring systems.

`osbeehives.com`

GLUCOMETER

Universal glucometer works with any test strips.

`hackaday.io/project/10865`

MOBILECG

Open source clinical grade heart monitoring.

`github.com/peterisza`

OPEN BIOMEDICAL

Open source 3D printable health and biomedical devices.

`openbiomedical.org`

ULTRASOUND DEVICE

Open source ultrasound hardware platform.

`un0rick.cc`

AIR CITIZEN

Air quality monitoring hardware and software.

`aircitizen.org`

OPEN APS

Open artificial pancreas system designed for diabetics.

`openaps.org`

F.LAB

3D-printed DIY science and lab equipment.

`progressth.org`

OPEN BRAILLE

Hardware and software for the visually impaired.

`github.com/carloscamposalcocer/OpenBraille`

FARMHACK

Open source farming tools.

`farmhack.org/tools`

FARMBOT

Automated farming gear.

`farmbot.io`

THE NODE MINI SERVER VERSION 3

Now that the Raspberry Pi 4 is out, we have an excellent candidate to base the mini server on. The faster processor and option for 8GB of RAM opens up more possibilities for P2P and self hosting applications, and is now more than capable to work as a general Linux desktop system too.

Like V2, this design is open source, and is based on a 3D printed frame topped with a PCB. This means you can style it however you want, with custom PCB and frame colours, and completely custom designs on the top cover.

The first thing you notice is the smaller size of the device. It's approximately 90x90mm and 25mm thick, which is only slightly bigger than the Pi 4 itself.

That makes it one of the smallest full-featured mini computers/servers out there, and is perfect for our needs.

The size and weight is also perfect for being mounted on the back of a monitor to be used as an all-in-one system.

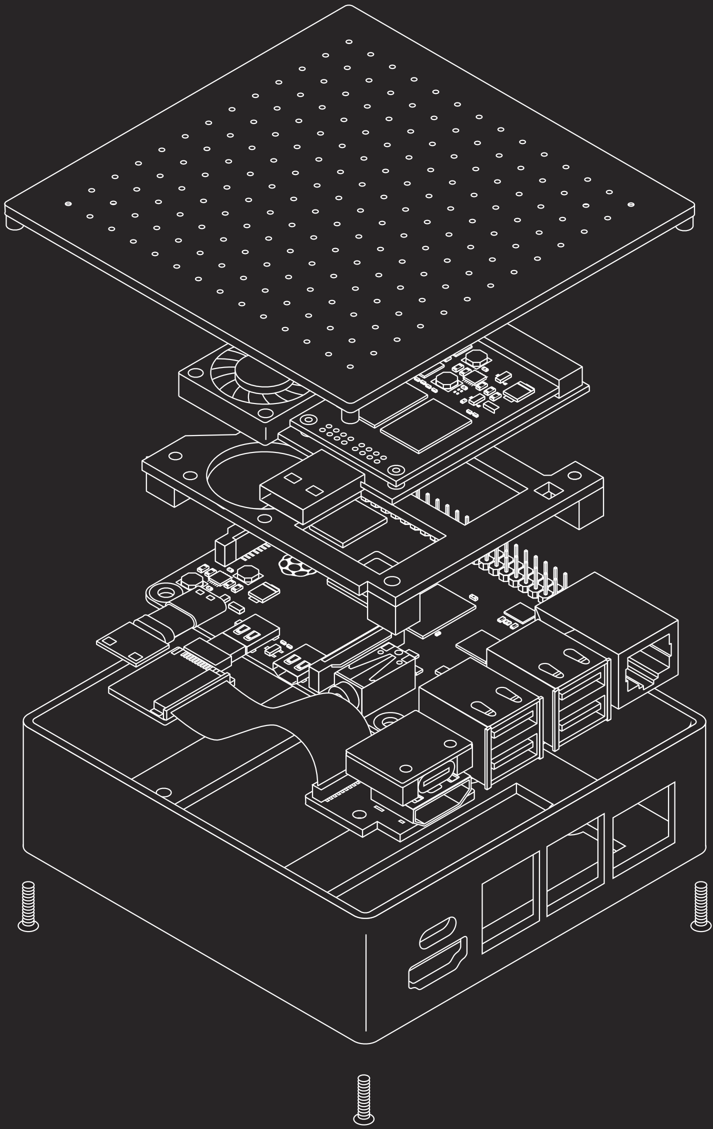
SPECS

- 1.5GHz Quad core ARM V8 CPU
- up to 8GB LPDDR4-3200 SDRAM
- 2.4 GHz and 5.0 GHz 802.11ac WiFi, BT 5.0
- Gigabit Ethernet
- 2 USB 3.0 ports
- 2 USB 2.0 ports
- mSATA SSD via USB 3.0 port (max 2TB)
- 5V DC via USB-C connector (min 3A)

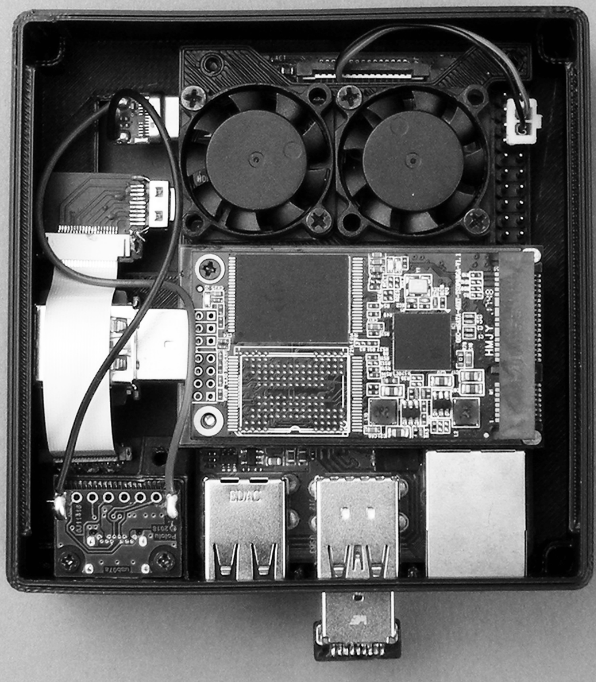
MODULAR

A main feature I worked hard on for this iteration is modularity, and many of you will be glad to know this version requires no modifications to the Pi itself.

I created a custom adapter HDMI and USB-C ports, leaving you with a selection of ports perfect for server use, and all located on the back side of the device.









INTERNAL SSD

Instead of a 2.5inch hard drive and a USB 2.0 adapter, Mini Server V3 now takes advantage of USB 3.0 speeds and the small size of an mSATA solid state drive.

I created a custom USB 3 extender, allowing you to simply plug in the mSATA SSD adapter into the Raspberry Pi 4.

The widely available USB 3.0 to mSATA adapters you can find on eBay, are the perfect size for this, with the screw holes on the Pi lining up perfectly to the adapter.

With this being modular too, you can forgo the SSD and just use the internal micro SD card if you wish. Both work fine.

HEAT MANAGEMENT

Heat is more of a challenge when combining the SSD and the Pi 4, so this time I decided to include two fans inside which run constantly when the server is turned on, providing a constant stream of air around the parts.

PARTS

- Raspberry Pi 4
- Top PCB (88x88mm 1.6mm thick)
- Male USB3 PCB (7x14mm 2mm thick)
- Female USB3 PCB (19x14mm 1.6mm thick)
- Full size HDMI PCB (21x24mm 1.6mm thick)
- Micro HDMI PCB (14x18mm 1mm)
- Pololu USB 2.0 Type-C Connector Breakout
- USB-C Male Plug Breakout Board
- Micro HDMI Male Component (Wedge Type)
- Male USB 3.0 Plug (692112030100)
- Female USB 3.0 Connector (48405-0003)
- 25x7mm 5V Fan (JST XH 2-Pin connector)
- 10pin 100mm 1mm Pitch Flex Cable
- 20pin 50mm 0.5mm Pitch Flex Cable
- USB3 to mSATA SSD Adapter
- mSATA Solid State Drive (up to 2TB)
- HDMI Type A Connector (47151-1001)
- 10pin 1mm Pitch Connector - (52271-1079)
- 20pin 0.5mm Pitch Connector (52746-2071)
- 20pin 0.5 Pitch Connector (20FLZT-SM1-TF)

(All PCBs are made from standard FR4 material, and are 2 layers. You'll be able to find the files for the boards and 3D printable parts on the NODE website)

Low Frequency RFID
125-134 KHz

Cordless Phone (AM)
1.7 MHz

Near Field Communication
13.56 MHz

Pager (Europe)
26.200 - 27.995 MHz

Pager (UK, Medical)
26.230 - 26.870 MHz

Cordless Phone (FM)
27 MHz

Pager (USA, Unlicensed)
27.255 MHz

Cordless Phone (FM)
43–50 MHz

Pager (USA, Emergency)
152.0075,157.45, 163.25 MHz

IOT Standard
169 MHz

Keyless Entry (USA)
315 MHz

LORA
433 MHz

Ultra High Frequency RFID
433, 860-956 MHz

Keyless Entry (Europe / Asia)
433.92 MHz

LORA (Europe)
868 MHz

Cordless Phone
900 MHz

LORA (USA)
915 MHz

Aircraft Tracking
1090 MHz

GPS
1176.45, 1227.6, 1575.42 MHz

Cell Phone (3G/GSM)
800, 850, 1700, 1900, 2100 MHz

Cell Phone (4G/LTE)
800, 850, 1900, 1700, 2100 MHz

Bluetooth
2.4-2.4835 GHz

Microwave RFID
2.45-5.8 GHz

WiFi (802.11b/g, a, n)
2.4, 5.0, 2.4/5.0 GHz

ISLANDS IN THE NET - THE ALOHANET

In the late 1960's, Norman Abramson had a feeling that connecting computers over the existing telephone network was a bad idea. Abramson, then a 36-year-old professor of both Electrical Engineering and Computer Science at the University of Hawaii, foresaw future data networking needs and didn't think the telephone system architecture was up to the challenge. It was time, he thought, to move to the airwaves.

At the time, the University of Hawaii had a total of seven campuses scattered around the islands. A main campus sat in Manoa, a four year school rested in Hilo, and five two year community colleges were placed in Oahu, Kauai, Maui, Hololulu, and Hilo (this is in addition to the four year school mentioned previously). Connecting campuses separated by water was no easy task and telephone lines were unreliable—clumsy and ineffective if a storm were to roll through and wreak havoc.

Using radio equipment to link computing resources together, Abramson speculated, could prove to be a good alternative in terms of reliability and control as UH would own the infrastructure. However, wireless networks like this were unheard of at the time; there wasn't a turn-key solution that could be implemented after a week or two of planning and installation. Abramson, with his newly-created experimental network team, would have to build the system themselves.

After a few months, Abramson's team decided on a random-access protocol for sending data through the network. The network itself would be structured in a star configuration with a "hub" located at the Manoa main campus, serving as both a topological and geographic center. From there, each additional campus (positioned up to about 100 km away) would act as a client, only communicating directly with the hub. Using a star model helped reduce overall costs within the network as only the hub needed to do the heavy lifting and clients could get away with a more modest set of resources. The network would also implement a two-channel UHF radio configuration, utilizing 100 KHz channels at

407.350 MHz and 413.475 MHz, chosen after the team received some advice from David Braverman of Hughes Aircraft Company. One channel would be devoted to client communication, so any node could send packets to the hub and only the hub would be listening. The second channel was for broadcast by the hub, which would send acknowledgments back on successful client transmissions. While each client would receive every acknowledgment, it would be discarded unless the transmission was specifically meant for the client listening—the client only cares about its own acknowledgments.

The random-access principle came into play with regard to how clients would use their communication channel. Due to how computer traffic often happens in bursts, clients transmit packets on the channel (dubbed the "ALOHA channel") to the hub as soon as they are available to go out. On successful transmission, the hub sends acknowledgment back on the broadcast channel. However, clients never check to see if the channel is in use before transmitting, so collisions can occur if the channel is already noisy from another client's traffic. To combat this, each client monitors their acknowledgment packets from the hub and will wait for a random amount of time before retransmitting if they don't hear anything back, indicating that their data was lost or garbled along the way.

The structure of this network was unlike anything else that was seen at the time for connecting remote computing sites, which often consisted of point-to-point telephone or microwave links that could only accommodate two connecting devices. Hardware for the campus sites was something that still needed consideration. How would all of these locations process and convey information to one or more peers? Team members Alan Okinaka and David Wax built a custom solution to be deployed at client sites: the Terminal Control Unit (TCU). Constructed out of an antenna, transceiver, modem, buffer, and control unit, the TCU was equipped with a standard RS232 interface to hook up to a terminal, allowing data rates of 9500 bits/s between client and hub. To save in hardware costs, TCUs were only capable of half-duplex communications—they could only transmit or receive at any one time, not both. The central communications hardware installed at the hub

was an HP 2100 minicomputer called the Menehune. In the Hawaiian language, Menehune translates to "imp," a direct reference to ARPANET's Interface Message Processor (IMP) which was being developed around the same time. Unlike the TCUs, Menehune was more robust and supported full-duplex communication to allow simultaneous transmission and receiving.

In June 1971, ALOHAnet (originally standing for Additive Links On-line Hawaii Area) was unveiled and became the first network to connect computers via radio technology. At the time, there was no group within the university set up to commercialize research successes, so the technology behind ALOHAnet was entered into the public domain.

Throughout the next year, the team would construct more TCUs to deploy to various campuses within the university and run the new network through its paces.

In the early 1970's, domestic satellites were becoming the next logical step in wireless network expansion, and Abramson's team was keen on working with the new technology.

At this point in time, ALOHAnet's funding had been switched to the Advanced Research Projects Agency (ARPA) under the direction of Lawrence Roberts, then known for his work with the ARPANET.

By 1973, ALOHA technology would soon be implemented in an experimental satellite network, the Pacific Educational Computer Network (PACNET), which connected NASA Ames Research Center in California, the University of Sydney, the University of Alaska, and Tohoku University in Sendai, Japan via the six-year-old ATS-1 satellite and low-cost ground stations. PACNET was a success, and the next step for ALOHAnet seemed to be, so the team thought, to connect it up to the ARPANET via satellite technology.

How the ALOHAnet was ultimately connected to the ARPANET is a humorous story by Abramson's account. In 1972, Abramson had a meeting at Roberts' stateside office to discuss the status of the ALOHAnet project. Roberts was called out of his office for a few minutes and Abramson happened to notice a list of planned IMP installation locations on a nearby blackboard for sites that would soon be added



to the growing ARPANET. Abramson was planning on bringing up the topic of connecting the ALOHAnet later, so he added "ALOHAnet" to the board with a randomly-chosen date, "December 17," written beside it. There wasn't a good chance to bring the topic up after Roberts returned to his office, and Abramson forgot about it completely himself until he received a call in early December from the IMP deployment team who asked him to prepare a location for installation. It seemed that Roberts either thought he added that entry to the board himself, or that it was an idea that didn't need any further discussion.

On December 17th of 1972, ALOHAnet was connected to the ARPANET, becoming the first site added by satellite link. Over the following years, ALOHANet refined its main protocol to create "Slotted ALOHA," which allows transmissions to occur only within certain timeslots. Several nodes can operate in a single timeslot, but if multiple nodes try to transmit at the same time, a collision occurs and they attempt transmission again in a later timeslot after a random delay. This helped greatly with network throughput by cutting down how many nodes could attempt transmission at a certain time, but didn't go so far as to allow one node exclusive ownership of a slot (which would slow down the network even more, as discovered with the development of Pure ALOHA years earlier). By 1974, with the introduction of the Intel 8080, ALOHAnet received a hardware upgrade as TCUs were replaced with new Programmable Control Units (PCU) that utilized the new 8-bit microprocessor. Now, network protocols could be written in software instead of built into hardware, meaning others could easily create their own packet radio data networks with readily-available chips. Packet radio networks would become a point of further experimentation by ARPA researchers, yielding some success with networks like PRNET in the San Francisco Bay area.

The connection to the ARPANET wasn't the only important event for ALOHA in 1972. In the same year Bob Metcalfe, a Xerox Palo Alto Research Center (PARC) employee and Harvard University graduate student, was visiting his friend and ARPANET protocol developer Steve Crocker while in Washington for a business trip. Crocker had previously met with Norm Abramson to discuss ALOHA, and

left a paper on the network out while Metcalfe was staying for the night. Finding the paper before going to sleep, Metcalfe read through it and was disappointed in mathematical assumptions made about the network's performance and capabilities.

He would soon travel to Hawaii and meet with the ALOHA team before returning to PARC and breaking ground on a networking protocol for their new Alto computer. Borrowing concepts from ALOHA, Metcalfe would create the "Alto Aloha Network," an experimental network utilizing cables to link Altos in a lab environment. The resulting network was a success, and over a thousand times faster than the wireless ALOHAnet. In a stroke of brilliance, Metcalfe would devise a new name for his technology: Ethernet.

As communications technologies evolved, ALOHAnet's usefulness began to decline. Still, many technologies that were invented for ALOHAnet were later successful in commercial use and are ingrained in many devices and products used today. Slotted ALOHA would go on to see use in satellite communications for both military and commercial use, as well as in set-top box communications, RFID technologies, and mobile telephony (even in 3G phones!). Carrier Sense Multiple Access (CSMA), a popular access control protocol used by Ethernet, WiFi, and other technologies, was directly inspired by ALOHA's random-access channel architecture. While ALOHAnet started as a simple experiment, it has forever shaped the way we construct networks, and ultimately, influenced the way we communicate with one another.

—

Islands in the Net is a column exploring ephemeral, decommissioned, or little-known communications networks. This first installment is only coincidentally about a network that was literally used to connect several islands together.

P2P LIVESTREAMING

One area of Internet activity that's exploded in popularity over the past few years is video live streaming, which in itself is a kind of decentralization of what previously used to be traditional broadcast media.

All of the main streaming services out there today are centralized, and to be fair there are advantages to that as live streaming is difficult to implement through P2P systems.

There are some projects out there working on just this challenge though, and I want to show them to you.

HYPERCAST

The first is hypercast by Louis Center. This super simple JavaScript app is a fork of another project called hypervision, and is incredibly easy to use. Hypercast runs on top of the Dat network, and requires the Beaker Browser (or other compatible browsers) to run.

To broadcast you simply need to clone the GitHub repo and follow the instructions to run the streaming server, you then press the [Start Broadcast] button in the browser. Following this, the stream will begin, and a `dat://` address will be generated which you can then share with your friends.

Anyone who goes to this address automatically begins seeing your stream. The data is shared around the connected users, so there's no need for centralized servers.

Be aware though, that since you're connecting directly to the other peers you'll all be able to see each others IP addresses (unless using a VPN or TOR).

Hypercast is currently very bare-bones, and automatically streams the default video and audio inputs on your system, but more options are planned, along with audio-only broadcasts, archiving, and chat.

`github.com/louiscenter/hypercast`


ON AIR
STOP BROADCAST
dat:/ /39Hj819hHd98J8S

IPFS LIVESTREAMING

If you want a more customizable streaming experience, you might want to check out the IPFS Live Streaming project (`github.com/tomeshnet/ipfs-live-streaming`). Set up by ASoTNetworks and darkdrgn2k of Toronto Mesh, it was originally designed specifically for streaming the Our Network 2018 conference, so it has a lot of extra features compared to Hypercast.

This method delivers something much more akin to what 'traditional' streamers would be used to. The example setup given by the team (shown on the next page) and can utilize multiple microphone inputs, and a mixture of HD camera, projector and desktop streaming inputs. This setup also supports a separate controller laptop which runs OBS (Open Broadcaster Software, `obsproject.com`), which can stream to both HTTP & IPFS at the same time, while also recording a backup of the stream to an external drive.

The installation for this one is much more involved, and has a lot of options, so you can check that out on their GitHub page.

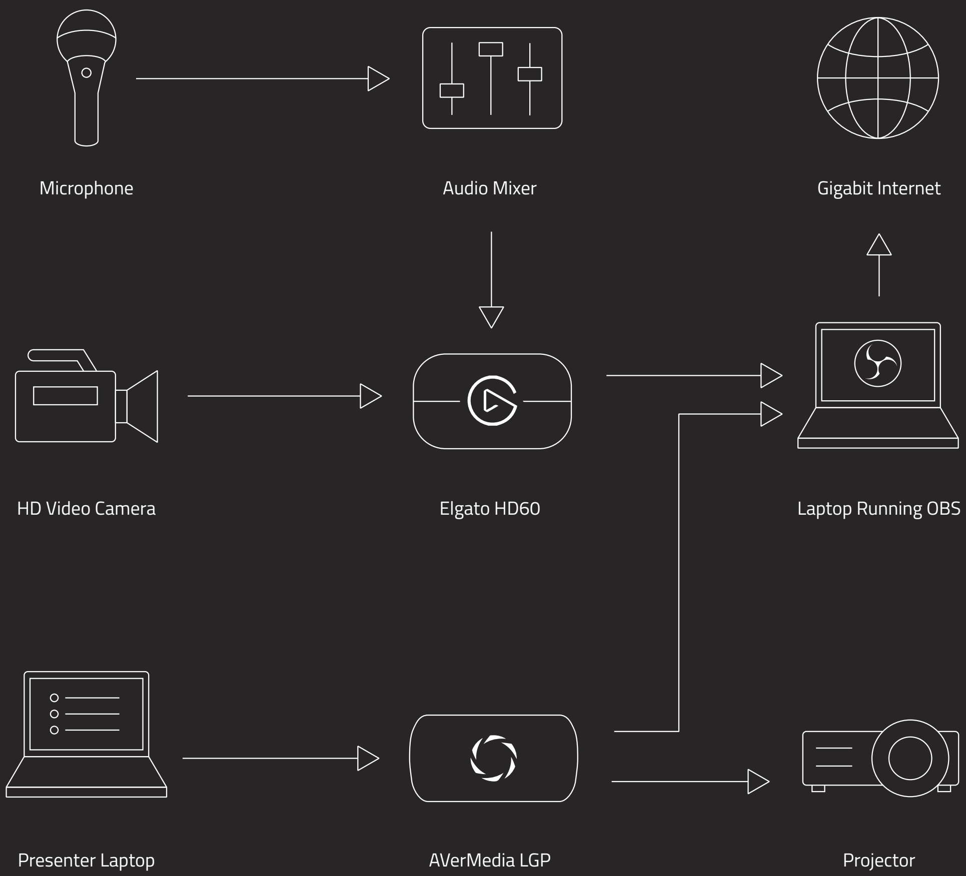
Once you start broadcasting, similar to hypercast, different URLs are generated (both for HTTPS and IPNS), and these can be used to watch the stream.

CONCLUSION

There's something really exciting and powerful about being able to set up a video livestream without needing to use any services other than a network connection.

These projects are both pretty young, but each delivers working software that can grow and improve over time.

If you have the skills to help, I'd recommend checking out each project's GitHub page and seeing if you can help out.



Microphone
Audio Mixer
Gigabit Internet
HD Video Camera
Elgato HD60
Laptop Running OBS
Presenter Laptop
AVerMedia LGP
Projector

CLEANING UP YOUR ONLINE FOOTPRINT

As the Internet continues to age, we're starting to realize just how much information about ourselves is floating around on random servers around the world, and how far back it all goes.

You might not care about that old blog post, or Myspace page that's still up, but maybe in the future you will.

This kind of information is not only important in terms of things like finding and keeping employment--because we've ALL said dumb shit one time or another--but also for more advanced phishing and social engineering. If an attacker can figure out details about you, what you're like, what your interests are, where you work, who you know, etc., then it's much easier for them to craft a perfectly designed attack to manipulate you.

The best rule is to not interact with the 'net so you don't leave tracks in the first place, but for normal people, here are a few tips you could try out to reduce your online footprint.

Remember that there are levels of effectiveness to this. Most of these won't do anything to stop the employees of tech companies or government agencies from knowing your info, because they probably already have searchable records already, but against random people on the net and potential attackers, some of this may help.

Search Yourself. The first thing to do is search yourself, and see what's out there. You can use multiple search engines and search for things like your name, physical address, email address, usernames you currently or have used, reverse search avatar pics and photos you've uploaded publicly.

Delete Old Accounts. If you find a bunch of old accounts that you haven't used in a long time, it's probably worth deleting them. `https://justdelete.me` will help you find instructions and links for deleting different accounts.

Edit Old Accounts. If you cannot delete these old accounts, the next best thing is to change all the info, contacts, and photos to random, unrelated stuff. This means anyone searching for you who stumbles across your old profiles will see completely different information. It's also worth reseting old account passwords to reduce the risk of hijacking due to potential breaches.

Change Privacy Settings. If you want to keep using certain social sites, but don't want random people searching everything about you, don't forget to change the privacy settings on these apps. Most of them have a private mode which stops random people checking out your profile, and it usually stops web crawlers, too.

Look at Cached Sites. Old links may be cached in different places, some of which can have removal policies. It may not work in all cases, but you may be able to contact the site owners to manually remove them.

Compartmentalize. Separating the different areas of your digital life is always a good idea. That means using different usernames, email addresses, and passwords for business and personal tasks, and never publicly linking these. This makes it much harder to find everything out if one area is compromised somehow.

Remove Metadata. If you're going to be uploading photos to social networks, one way to reduce some of the tracking and identifying information is to remove EXIF metadata. This makes it harder to tie things like usage / location patterns to you, which could in-turn be used by social engineers to attack you.

Don’t Post. I know in this age of nonsense, there's a compulsion to tell everyone about every little detail of your life, but if you don't need to--and you probably don't--I would resist the urge. This is the best way to keep your digital footprint clean.

LIBREROUTER - AN INTERVIEW WITH NICOLAS PACE

Starting in 2015, the LibreRouter project aims to address the needs of community networks by creating a hardware solution meant specifically for mesh. All over the world, community networks are thriving, but they are often deploying off-the-shelf hardware that was not intended for this purpose.

The LibreRouter aims to be an open platform robust enough to withstand the elements, poor line-of-sight, and limited power options. It will keep the network running without worry or constant tinkering. Coupled with the open-source LibreMesh firmware (based on OpenWRT), the LibreRouter is built to be the de facto device for community networks.

Nicolas Pace is a representative for LibreRouter, but he has additional experience in community networks. Pace is also a member of AlterMundi (`altermundi.net`).

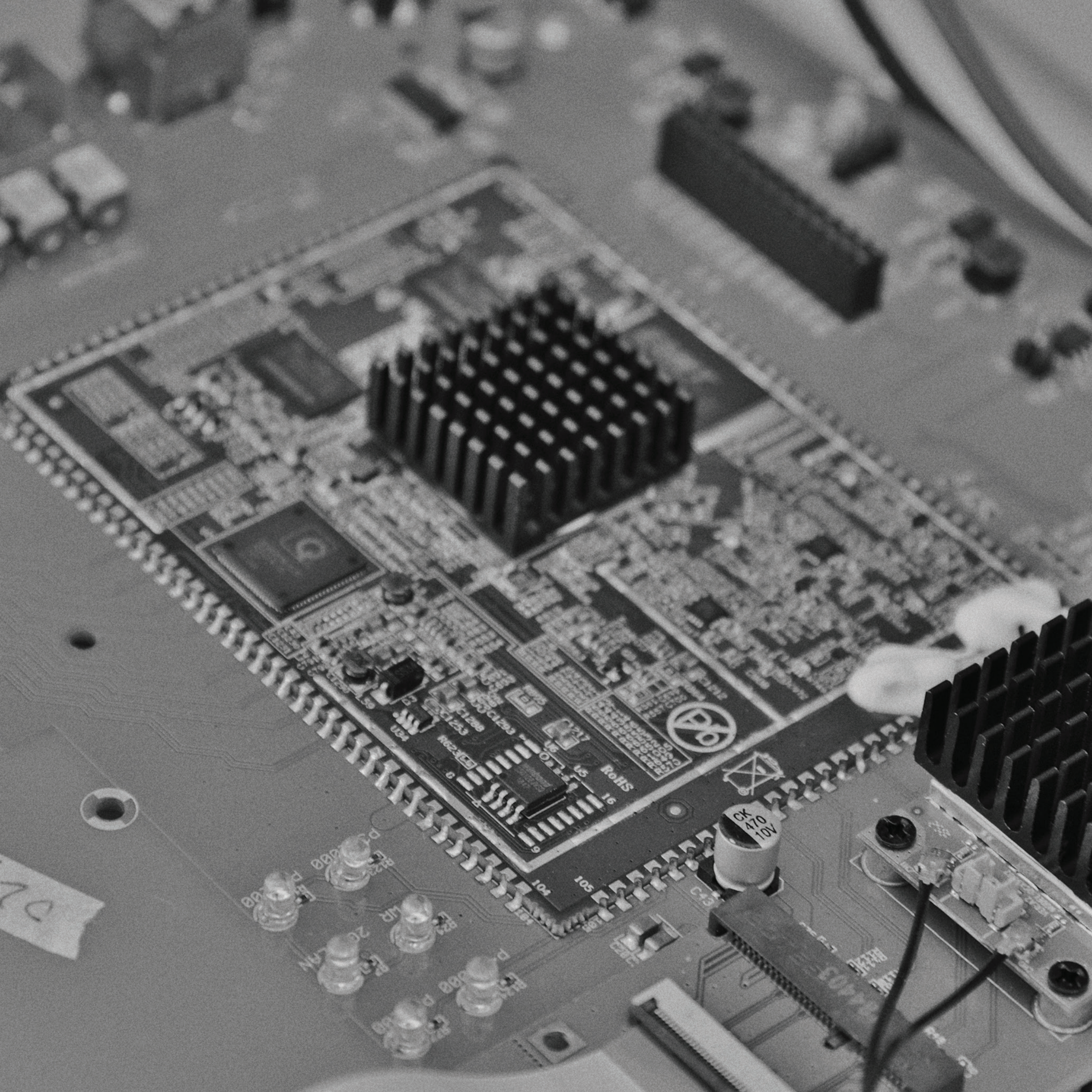
What are some good reasons to choose the LibreRouter over off-the-shelf hardware when building community networks?

The need of the LibreRouter comes out of the frustration that is using off-the-shelf hardware for building community networks.

For years the community networks movement has been using off-the-shelf wireless equipment for their networks (Guifi.net, Freifunk, AlterMundi, SudoMesh, Funkfeuer, among others)... but not without issues...

If any community would use them "as they are," with the stock software, a high learning curve is expected and there will be limitations as per how they could be used. Basically, they should be used as the manufacturer intended them to be. For example, this infrastructure is thought to be managed by a few, [and] centralized, so it doesn't promote decentralized management. It doesn't promote local innovation, as it doesn't allow tinkering with it. Also, it doesn't allow to locally fix issues that the device might have related to security or bugs that the manufacturer might not even repair.




RoHS
CK
470
10V

On the other hand, if you want to modify its factory behaviour to do other stuff that the manufacturer didn't design it for, prepare yourself as it will be a bumpy ride. These devices haven't been designed to be repurposed, so a lot of useful information is missing (no blueprints, manuals, or source code [is] available) and the action of reusing them is usually seen as shady (though legal).

Still, in general the devices that are available in the market don't have the features required for implementing community networks so communities end up spending a lot of resources adapting solutions that sort of somehow partially solve the problem, but they don't... so they find their way around it.

The LibreRouter is a stable and proven platform for deploying community networks at scale. It provides a geek-free approach too, so anyone with no prior technical skills can learn in 30 minutes to two hours how to set it up, expand it, and maintain it.

It is also a technology that builds community through building a network, [uniting] people , and encouraging collaboration.

How open is the hardware/software?

It is as open as it can be. The hardware designs are being ported to KiCAD as we speak (we used Eagle for the initial design because it was the software the factory was most used to, but decided to migrate to KiCAD so we can use Open Source tools fully too).

The software is Open Source, and is available at `github.com/librerouterorg`

Is the router free of binary blobs?

It is free of binary blobs. We still use chipsets that are closed source (Qualcomm CPU, Qualcomm Switch controller, Qualcomm Wireless radios), but they are the most open available right now. We have also chosen radios that allow for innovation, they have SoftMAC stack, meaning you can play around with how the radio treats the data.

Is it easy to modify and debug?

The build process is pretty straightforward, the same as OpenWRT (so any OpenWRT developer can start right away).

Debugging it can be a little challenging. The LibreMesh architecture is not typical, so you need to understand how it works in order to find yourself around the codebase, and as it is an embedded system it has its [own challenges] in itself.

It has all the debugging features available (serial interface, JTAG, unlocked boot partition).

Some networks are already making use of LibreRouters, how are they performing?

The first LibreRouter prototypes have been in the wild for two years approximately.

The first communities that are using it are in Colombia, Brazil, Mexico and India, and though they have not been very public about it, they are very happy about how it performs.

Something important to note is that the LibreRouter project is much more than just the router, it is an ecosystem for geek-free community networks that include the router, software, documentation, a support platform, a learning community, everything in as many languages as exist, with innovations in spectrum to get the best for the most needed areas. We are just at the beginning of a long journey, that already looks promising!

How flexible are the LibreRouter and LibreMesh software? Can I run LibreMesh on my existing OpenWRT devices and have them network with LibreRouters?

LibreMesh is a repository on top of OpenWRT so wherever you can run OpenWRT, you can run LibreMesh.

The reason why the LibreRouter exists though is because we want the communities to have a successful out-of-the-box experience when deploying community networks, and in order for that to happen we want to make sure they have everything they need:

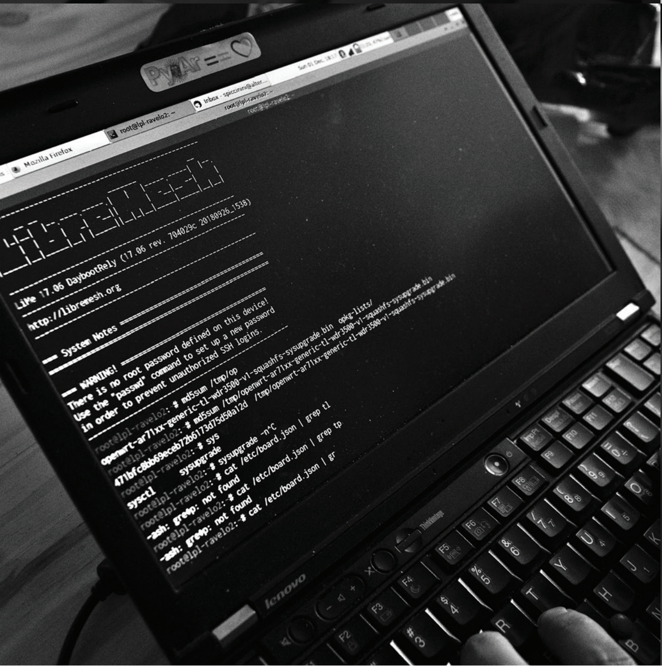
- Hardware that is reliable, well documented, open, and well supported.
- A deployment process that is predictable, and non-technical.
- A network that makes sense for them, that they can understand and expand with no prior knowledge.

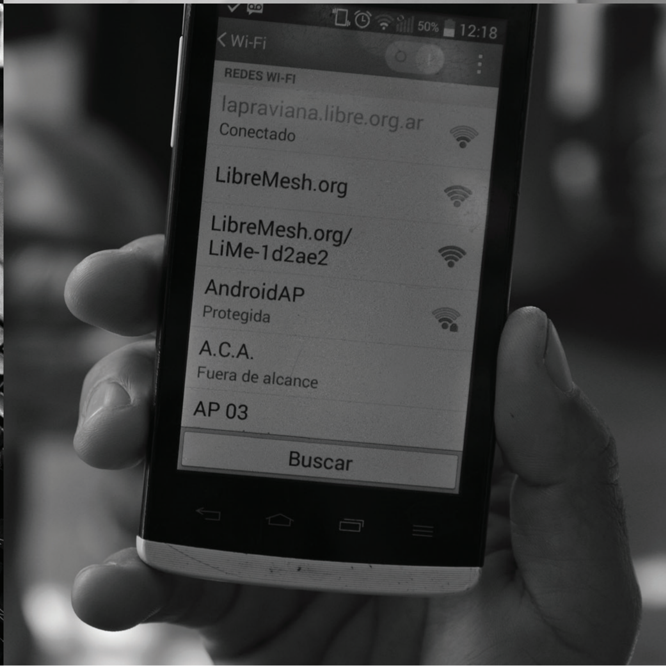


Setting up a community network


router




PyAr
Mozilla Firefox
LiMe 17.06 DaybootRely (17.06 rev. 704029c 20180926_1538)
http://libremesh.org
System Notes
WARNING!
There is no root password defined on this device!
Use the "passwd" command to set up a new password
in order to prevent unauthorized SSH logins.
lenovo


50% 12:18
Wi-Fi
REDES WI-FI
lapraviana.libre.org.ar
Conectado
LibreMesh.org
LibreMesh.org/
LiMe-1d2ae2
AndroidAP
Protegida
A.C.A.
Fuera de alcance
AP 03
Buscar

LibreMesh works best on the LibreRouter, and all of the features LibreMesh provides will be supported on it (and because it is the only option with all the hardware required for it, it will be the best LibreMesh experience of all).

As the LibreRouter itself does not only come with LibreMesh (and thus with OpenWRT) from factory, but also is an open source hardware router, the software can be changed to whatever you want.

Do you recommend specific external antennas that work well with the router?

The LibreRouter comes with three antennas! Two 5ghz sector antenna 15dbi MIMO 2x2, and one 2.4ghz sector antenna.

The best antennas we have right now are a clone of the PowerBeam dish with two dipole antennas as feeder.

We have done videos and tutorials on how to build them, you can check it at here:
`youtu.be/_3LyuF2qSSY`
`youtu.be/vjGvCFJJjRQ`

When can we expect the LibreRouter to be available for sale and is there an expected price point?

The LibreRouter is becoming more and more available as we become more capable of delivering. If you are part of a community that wants to use it, please let us know at `info@altermundi.net`

The price that we expect it to have all around the globe is ~170USD including taxes and shipping. It is hard to predict these costs across the globe, so think of this value as a reference.

Thanks Nicolas. More information can be found at the LibreRouter website: `librerouter.org`

DIGITAL ROT

I'm one of the only people I know who still owns a flatbed scanner. It's thin, black, sleek, but still makes that humming, whirring noise when you turn it on so it can warm up. Back when I was in school, I'd use it to scan books I procured from the interlibrary loan service offered by the university library. I could request rare books from libraries all over, scan them between classes over a few days, and assemble digital files to peruse later.

These days, I primarily use the scanner for digitizing old magazines and books. This is the stuff you can't really find on the Internet. Sure, maybe there will be an issue of something for sale on eBay every now and then to support proof of existence beyond a sparse Wikipedia article, but a lot of the publications I get my hands on are largely forgotten. Articles are locked away from the world in these paper safes until somebody pries one open and exposes what they find.

I share my scans online; I don't believe in keeping things in silos. I call this whole process preservation, but is it really? We have intact, readable books from hundreds of years ago, but digital files from only a few decades ago can be inaccessible, encoded in an unreadable format or suffer from data corruption and bit rot. We see a lot of issues like this now when it comes to saving floppy disks. There are maybe a dozen hardware devices out there designed to back up disks formatted for various computers or through different archival methods.

The Commodore 64 stored data differently than the Apple //, and you need to take that into account to get an accurate copy. You can make a bit-by-bit copy of a disk or read the entire magnetic flux, but after you actually get access to the files, you might find them to be in a format that nothing will read. Or worse, they might be corrupted as the media they're on has degraded over time.

We take the life of a lot of our media for granted. Floppy disks and magnetic tapes are thought to last around 20 years before having issues. Burned CDs and DVDs are good for



about 10 years (pressed media is closer to 30), along with solid state drives and flash media. Hard drives and burned Blu-Rays are less optimistic with expectancies of between two and five years. Survivorship bias may come into play if you think these times are more than a little pessimistic. I have 30-year-old floppies, burned CDs pushing 15, and hard drives at 20+ years that are still functioning and usable. The unfortunate truth is that these are all on borrowed time. I can't count on them for archival storage, and ideally I'd transfer files to new media every few years before the current vessel decays.

FORMAT	**LIFESPAN**
Floppy Disk	*20 years*
Magnetic Tape	*20 years*
CD / DVD	*30 years*
CD-R / DVD-R	*10 years*
Solid State Drive	*10 years*
Flash Memory	*10 years*
Hard Disk Drive	*2-5 years*
BD	*100 years*
BD-R	*2-5 years*

While physical media gets more difficult to preserve every year, I worry more about our native digital media. People are constantly producing content these days, whether it be music on SoundCloud or videos on YouTube or podcasts on Podbean or blog posts on Medium. Who is to say how many of these companies or products will be around in ten years and whether they will give an opportunity for users to export data if they're circling the drain. What about the people who would get there too late or aren't alive anymore as we saw with the shutdown of GeoCities? Do you trust YouTube to handle your videos or Instagram to keep an eye on your photos for you? If you don't keep a copy of your data, you run the chance of losing it.

You've likely run into digital rot since you first started using the Internet. Dead links have always been a problem when it comes to accessing content, and was actually a point of consideration in the early days of the Internet's design. The Internet was always designed to be decentralized, and because of this, it would always be a little bit broken. There isn't any centralized authority keeping information online, or notifying systems

owners when a host goes down so links can be updated. The Internet is much more of an organic system, with portions dying off as new entries grow and take their place. There is something beautiful about the chaos that goes into creating the Internet; it feels perfect knowing that it can't be perfect. That said, anyone doing research and stumbling upon what they think may be the perfect link has felt the torment of being redirected to a parked domain page. Ideally, good information is shared and spread to multiple sites on the Internet over time, but at the end of the day a lot of that disbursement relies on amateur SysOps and volunteers. People lose interest or something else comes up. Some sites are born, live, and die in relative silence.

While there is some beauty in the Internet's constant state of change, a lot of issues can arise as people treat it more and more as a form of permanence. 30 years ago you may have had a diary or a family photo album, but now people turn to websites for the same function. People put their trust into these services to keep their data safe and available, but it's a bit like handing your keepsakes to some company far away that you can't visit, and you can only hope they stay in business and don't lose track of anything. MySpace lost millions of MP3 files in 2019. Workshops for the computer-illiterate teach people to move all of their photos and documents to the cloud so they don't have to worry about finding things on their desktops or suffer losses from hard drive faults. The problem just gets a change of scenery.

Commercial content, not just user-created, is also in a concerning state. Providers like Netflix or Hulu really aren't producing a physical catalog of their shows and movies, a lot of which have the production value of major motion pictures that get theater releases. Over the last 50 years, we've gone through a cycle where little physical video content was made available to consumers, then seemingly all video content was made available, and now very little again. Do you expect Netflix to be around in ten years? Okay, how about 20? Sure, some other companies might buy up parts of their catalog upon a demise, but do you think you'll be able to see every television show available now after 20 years of company death and restructuring?

Will you have access to your digital video game library in 20 years? Will modern gaming consoles simply become useless pieces of hardware, orphaned from online "always on" networks that they depend on to run content locally? Will subscription based software cause a slow death for vintage computer communities?

Since the '80s, there has been a back-and-forth between commercial businesses and groups of hackers, crackers, and pirates. Copy protection on floppy disks, Macrovision on VHS tapes, and even more modern DRM on audio, video, and gaming content poses issues for the longevity of media as it ages.

Whether they know it or not, people cracking software and pirating content are archivists. Because of work done by the minority, we can physically store media that would otherwise be inaccessible. We can save it for later. We can worry a little less about it disappearing overnight. If you can't physically point at a piece of media, whether it be a game on a floppy disk or a movie stored digitally on a hard drive, you don't really "own" it.

If there is content out there that you want saved, the only way to be sure it is preserved is with citizen archiving. Learn how to use wget to make website dumps, or youtube-dl to save YouTube channels. Follow the 3-2-1 rule to save three copies of your data, with two of them using different storage media, and one of them kept offsite. Check your backups, make sure it isn't too late if they happen to fail. There is a lot of optimism with emerging storage technologies, like holographic discs and DNA, but if data isn't kept safe until these technologies are mainstream, it may be gone forever.

We've lost a lot of data, but hopefully we can get better at saving it.



UPGRADING THE THINKPAD X200

In the last issue, I showed you how to install Libreboot BIOS firmware on the ThinkPad X200, turning it into one of the freest laptops you could own right now.

Though it's perfectly fine for everyday usage (web browsing, non-intensive programs, and even light editing), in some ways it's a little outdated. That's why I want to show you how to upgrade it in various ways.

Older laptops didn't usually aim to be the thinnest devices possible, which is good for us, because it gives us space to add different mods and extras directly inside of the case.

1. PCMCIA SLOT

The X200 has a standard 54mm-wide PCMCIA or PC Card Express slot, giving us some expansion options (and space) inside.

These old ports also accept thinner 34mm cards, and that's exactly what I went for.
I added 2x USB 3.0 ports, for better connectivity and transfer speeds.

The 20mm space that was saved by using the thinner card also gave me some extra room to add a custom 3D printed insert into the slot specifically for micro SD card storage.

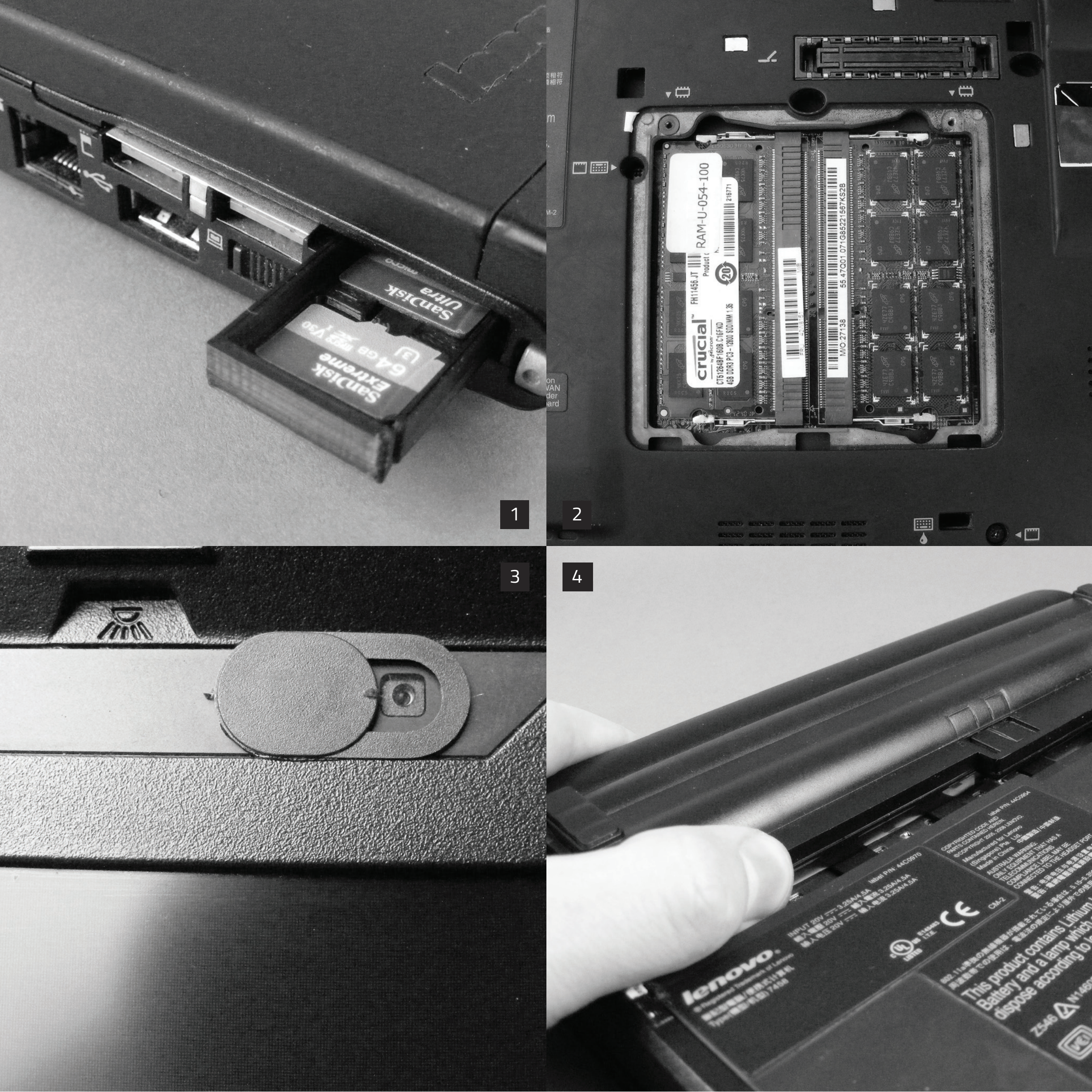
The idea here is to have a few different micro SD cards with live operating systems like TAILS or something with hard drive cloning utilities that I could always have on hand and use with the X200's built in card reader.

2. UPGRADING THE RAM

The official specs state that the max RAM capacity on the X200 is 4GB, but it actually extends to 8GB, giving you a fair chunk of memory to play with for more intensive tasks.

According to other X200 users, the machine is a little temperamental with which RAM sticks it accepts, so do your research. I used 2x Crucial 'CT51264BF160B' 4 gig sticks.


lenovo
X200
ThinkPad


1
SanDisk Ultra
SanDisk Extreme
64GB
crucial
RAM-U-054-100
2
3
4
lenovo

3. WEBCAM SLIDER STICKER

A standard addition to any laptop is the webcam slider sticker. Whatever you're security practices, it's always worth adding an extra physical layer to interfaces for manual control, just in case...

4.UPGRADING THE BATTERY

Another thing to replace is the standard capacity battery with its larger 9-cell alternative. This beast gives you 7800mAh performance, which can stretch to almost 6-7 hours battery life. Pretty remarkable for an ancient machine.

5. ADDING AN SSD

Another easy upgrade is to swap out the HDD with an SSD, for faster booting and overall performance. Another interesting idea is to use an mSATA SSD to SATA adapter board, and this gives you a load of extra space directly in the drive bay that could be used to hook up other custom components internally.

6. REMOVE INTERNAL WIFI CARD

If we remove the internal WiFi/Bluetooth card, we not only get a bit more space inside for other mods, but we can replace these interfaces with mini USB dongle versions.

This gives us a bit better security as we now have physical access over the connection, meaning when it's unplugged the machine is automatically (and verifiably) air-gapped.

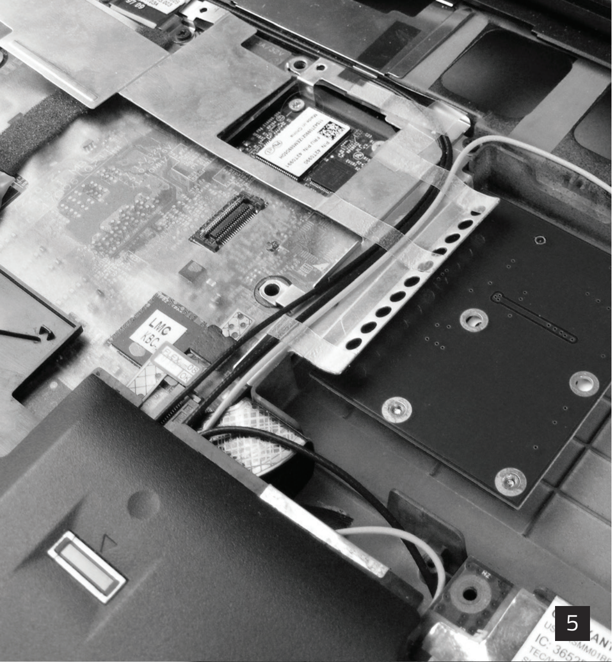
HONORABLE MENTIONS

There are more mods you can make, which I didn't get around to, but you may find useful.

The first is to replace the stock LCD panel with the "HV121WX4-120" model. This doesn't actually increase the 1280x800 resolution, but nonetheless is regarded as a superior display by X200 users due to it's more vibrant colours and wider viewing angles.

There's also a free Mini PCIe space next to the internal WiFi card, where you can install the "F3507G" WWAN modem. This links up to the




5



6

internal antennas inside the X200, and the SIM card slot (underneath the battery), to give you 3G and GPS connectivity. If you really wanted to go all out, you could create a custom adapter that adds a hardware switch, so you control when it's turned on and off.

CONCLUSION

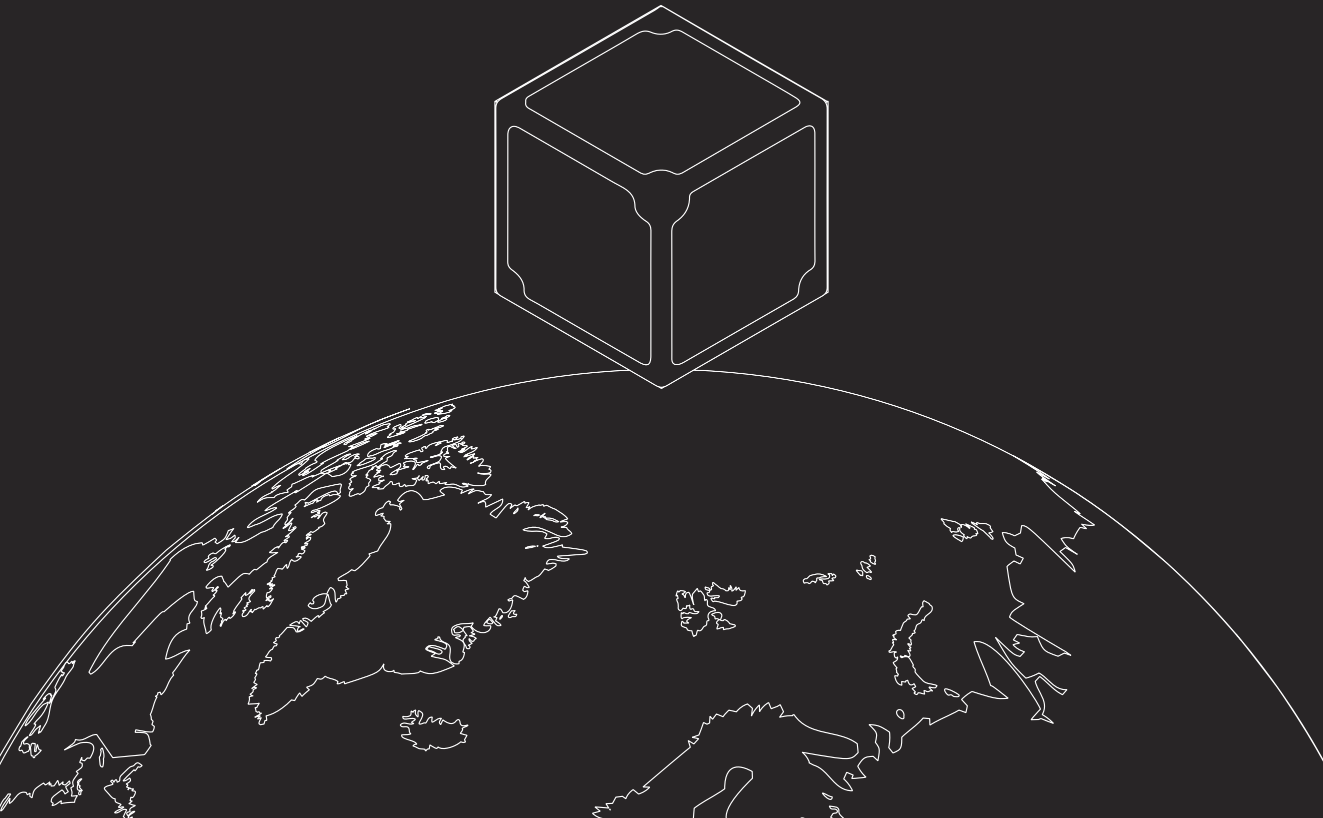
This isn't going to smash any benchmark tests, but that's not the point. What other Linux system costs a couple hundred dollars, and gives you this many interface options, a free software BIOS, a CPU without backdoors (that we know of), great keyboard design, 8GB RAM, and all-day battery life?

This upgraded system can handle most tasks outside of heavy editing and gaming, and for many, that's enough. I know I'm very happy with how this turned out.

There's a reason these older ThinkPad's are considered true workhorses. They're built to last, and I suspect their sturdy designs will be around for a long time to come.

IPFS 101

THE INTERPLANETARY FILE SYSTEM



InterPlanetary File System, more commonly known as IPFS, is an open-source protocol and network to store and share hypermedia in a peer-to-peer distributed file system. First released in 2014, IPFS combines concepts of distributed hash tables (or DHT), BitTorrent, and Git to create an alternative to HTTP, which currently dominates the web.

HOW IT WORKS

Two important design features of IPFS are that it uses DHT for peer-to-peer networking and Merkle DAGs to store data within the network.

IPFS is most commonly used by installing the official IPFS application, and invoking it with commands through your terminal. You can find download links for pre-built IPFS packages through the official website at `ipfs.io`.

After installing IPFS, here is how you can interact with it and utilize the IPFS network.

1) Opening up a terminal, first initialize IPFS for your user account:

```
$ ipfs init

initializing ipfs node at
/Users/NODE/.go-ipfs

generating 2048-bit RSA
keypair...done

peer identity:
Qmcpo2iLBikrdf1d6QU6vXuNb6P7hwrb...

to get started, enter:

ipfs cat
/ipfs/QmYwAPJzv5CZsnA625s3Xf2nemtYg
PpHdWEz79ojWnPbdG/readme
```

2) Behind the scenes, IPFS is creating a file repository in your home directory at `~/.ipfs` and a peer identity for your machine.

3) To get started, there is an introductory file available via the IPFS protocol with this hash and file name:

```
QmYwAPJzv5CZsnA625s3Xf2nemtYgPpHdWE
z79ojWnPbdG

readme
```

4) You can access the contents of the `'readme'` file using the `cat` command as shown here:

```
ipfs cat
/ipfs/QmYwAPJzv5CZsnA625s3Xf2nemtYg
PpHdWEz79ojWnPbdG/readme
```

5) This file is only read locally on your system, meaning you aren't on the IPFS network yet. Join the network by invoking the IPFS daemon:

```
$ ipfs daemon

Initializing daemon...

API server listening on
/ip4/127.0.0.1/tcp/5001

Gateway server listening on
/ip4/127.0.0.1/tcp/8080
```

6) Behind the scenes, IPFS will connect to a trusted default list of bootstrap peers which exist within the IPFS swarm.

These peers will be used to learn about others on the network when downloading files.

7) Suppose that a friend Bob wants to send you a picture of his cat through IPFS, and provides you with an address to the file:

```
/ipfs/QmW2WQi7j6c7UgJTarActp7tDNikE
4B2qXtFCfLPdsgaTQ/cat.jpg
```

8) Now that you are on the IPFS network, you can issue a command to download it:

```
ipfs cat
/ipfs/QmW2WQi7j6c7UgJTarActp7tDNikE
4B2qXtFCfLPdsgaTQ/cat.jpg > cat.jpg
```

9) Behind the scenes, IPFS will check with its peers to see if any of them have a file available matching this hash and file name. If any of them do have it, it will be downloaded, but if not, IPFS will leverage DHT to explore the network outward until it finds a node it can retrieve the file from.

Files are stored in IPFS as objects called Merkle DAGs, which are tree-like structures containing chunks of data chained together. When IPFS finds a node to download Bob's cat picture from, it must follow the links in the initial object it downloads until it retrieves every fragment to assemble the entire file.

10) Now let's suppose that you want to host a copy of Bob's photo, or any other file you happen to download via IPFS. This can be done by "pinning" the hash of the data you have downloaded, which will store it locally on your machine and make it available to others:

```
$ ipfs pin add
/ipfs/QmW2WQi7j6c7UgJTarActp7tDNikE
4B2qXtFCfLPdsgaTQ
```

11) You may also want to host your own file on the IPFS network. This will make the file available to everyone as long as it is still hosted on your machine, with the potential for it to last even longer than that if another node decides to pin it. Here we see how to host a text file titled `mytextfile.txt`:

```
$ echo "this is a sample file" >
mytextfile.txt

$ ipfs add mytextfile.txt

added
Qme9chBZHSH3njh34FPZKWn7vQVcmL4nPVJ
mED3q9be6RE mytextfile.txt
```

While most IPFS interfacing is commonly done via the terminal, running the IPFS daemon also provides you with a local gateway to access files via your browser or a Linux utility like `curl`. With `ipfs` running, you could load Bob's cat picture in your browser by visiting:

```
http://localhost:8080/ipfs/QmW2WQi7
j6c7UgJTarActp7tDNikE4B2qXtFCfLPdsg
aTQ/cat.jpg
```

To retrieve files on the IPFS network from a machine without the software installed, the IPFS developers have a publicly accessible gateway at `https://ipfs.io`.

This means you could easily retrieve `cat.jpg` from the following link, from anywhere with Internet access:

```
ipfs.io/ipfs/QmW2WQi7j6c7UgJTarActp
7tDNikE4B2qXtFCfLPdsgaTQ/cat.jpg
```

While this illustrates downloading and sharing single files, IPFS can also be used to share entire directories. This is exciting as IPFS has some more advanced mechanisms to assist addressing data that may be updated or otherwise changed. You could host a website within IPFS and update it at any time while users can reference it from the same address.

COMPARISONS

IPFS has some main differences from other distributed storage platforms such as Freenet, Storj, or MaidSafe.

Continued ⟶

IPFS allows for selective mirroring of data. Any given node does not need to mirror the entire contents of the network, or an arbitrary slice of data containing content the user doesn't want.

Additionally, IPFS is not bound to any cryptocurrency. Unlike Storj and Maidsafe, users are not required to pay a sum to have their files on the network, and there is no built-in incentivization for nodes storing files.

This, however, means that all mirroring in IPFS is done voluntarily. There is a related project called Filecoin which, much like Storj or Maidsafe, incentivizes file storage on IPFS with a new cryptocurrency.

DOWNSIDES

The major downsides of IPFS stem from its relatively low adoption rate when compared to the Internet or other P2P storage networks.

Content discovery is limited as current IPFS search engines are rudimentary, and file addresses are more cryptic than those used for HTTP resources.

`IPFS Search -`
`http://ipfs-search.com`

`ipfs-search -`
`https://ipfs.io/ipfs/QmYo5ZWqNW4ib1Ck4zdm6EKteX3zZWw1j4CVfKtnAzNdvu/`

	FREENET	**STORJ**	**MAIDSAFE**	**IPFS**
Design	*Decentralized*	*Decentralized*	*Decentralized*	*Decentralized*
Encryption	*Yes*	*Yes*	*Yes*	*Yes**
Censorship	*Resistant*	*Resistant*	*Resistant*	*Resistant*
Proof System	*None*	*Proof of Storage*	*Proof of Resource*	*Proof of Copy**
Compensation	*No*	*STORJ Token*	*Safecoin*	*Filecoin Token**
Blockchain	*No*	*Counterparty BTC*	*No*	*Yes**

** With Filecoin*

Additionally, browser support is currently lacking. Mainstream browsers do not currently have native access for content hosted within IPFS. There are however extensions for Chrome, Firefox, and Brave that allow IPFS access, while Brave developers are currently working on including IPFS support natively. The Beaker browser had IPFS support until 2017, and may consider adding it back sometime in the future.

It is also important to remember that files hosted on nodes are relatively public. This means that files made available to the network will inherently be tied to the IP address of the machine they can be downloaded from.

CONCLUSION

IPFS offers a unique step forward in the storage and delivery of data across the web.

In the future, we can look forward to better support of the protocol through our web browsers, as well as widespread use in applications utilizing it for data storage.

There are many interesting projects out there already using IPFS right now, including (but not limited to):

Decentralized online marketplace, OpenBazaar (`https://openbazaar.org`)

Cryptocurrency bounty hunting platform **Bounty0x** (`https://bounty0x.io`)

Decentraland, P2P virtual reality metaverse (`https://decentraland.org`)

Ethereum-based P2P social network, **AKASHA** (`https://akasha.world`)

DLive video livestreaming service (`https://dlive.tv`)

Music and artist discovery platform, **Audius** (`https://audius.co`)

To find out more about IPFS, check out the project's website - `https://ipfs.io`

MESHNET
ATLAS



Pretoria Wireless Users Group
Pretoria, South Africa
`https://ptawug.za.net`

Cape Town Wireless User Group
Cape Town, South Africa
`https://ctwug.za.net`

Durban Wireless Community
Durban, South Africa
`http://dwc.za.net`

Zenzeleni.net
Eastern Cape, South Africa
`http://zenzeleni.net`

Village Telco
Johannesburg, South Africa
`https://villagetelco.org`

Mesh Bukavu
Bukavu, DRC
`http://meshbukavu.org`

Redhook Wifi
Brooklyn, NY
https://redhookwifi.org

NYCMesh
New York City, NY
https://nycmesh.net

PittMesh
Pittsburgh, PA
https://pittmesh.net

Philly Mesh
Philadelphia, PA
https://phillymesh.net

PersonalTelco
Portland, OR
https://personaltelco.net

La Cañada Wireless Association
Santa Fe, NM
https://lcwireless.net

People's Open
Oakland, CA
https://peoplesopen.net

TFA Wireless
Huston, TX
https://techforall.org

CassCo WiFi
Detroit, MI
https://alliedmedia.org

Wasabinet
St. Louis, MO
http://gowasabi.net

Boston Meshnet
Boston, MA
https://bostonmeshnet.github.io

MassMesh
Boston, MA
https://massmesh.org

Toronoto Mesh
Toronto, Canada
https://tomesh.net

ZAP Sherbrooke
Sherbrooke, Canada
http://zapsherbrooke.org



Melbourne Wireless

Melbourne, Australia

`melbournewireless.org.au`

Air-Stream

Adelaide, Australia

`https://air-stream.org`





Network Bogota
Bogota, Colombia
`https://networkbogota.org`

Coolab
Throughout Brazil
`https://coolab.org`

Altermundi
Throughout Argentina
`https://altermundi.net`

FunkFeuer
Vienna, Austria
`https://funkfeuer.at`

Freifunk Leipzip
Leipzip, Germany
`https://freifunk.net`

Freifunk Berlin
Berlin, Germany
`https://freifunk.net`

pjodd
Malmö, Sweden
`https://pjodd.se`

Southampton Open Wireless Network
Southampton, UK
`https://sown.org.uk`

Guifi.net
Iberian peninsula, Spain
`http://guifi.net`

Ninux
Across Italy
`https://ninux.org`

Neco
Vietri di Potenza, Italy
`http://progettoneco.org`

AWMN
Athens, Greece
`http://awmn.net`

Wireless Leiden
Leiden, Netherlands
`https://wirelessleiden.nl`

WirelessPT
Moitas Venda, Portugal
`https://wirelesspt.net`

Wlan Slovenija
Ljubljana, Slovenia
`https://wlan-si.net`

Sarantaporo.gr
Elassona, Greece
`http://sarantaporo.gr`

Freemesh Denmark
Across Denmark
`https://freemesh.dk`



CABAL: THE P2P SLACK & IRC CHAT ALTERNATIVE

Cabal is a fairly new and experimental group chat app that takes aim squarely at Slack and IRC by offering a peer-to-peer alternative that is easy to use. I don't know too much about the team, but there seems to be a few Dat Project and Scuttlebutt members in there, which is really promising.

One of the interesting features of the app's P2P nature is that it works offline and locally, with chat histories and all the networking syncing automatically, wherever it's used. I can imagine that being particularly useful on meshnets and other local networks.

If you want to check out more of the technical details, and perhaps contribute to this fledgling project, check out their GitHub page:

`https://github.com/cabal-club`

USING CABAL

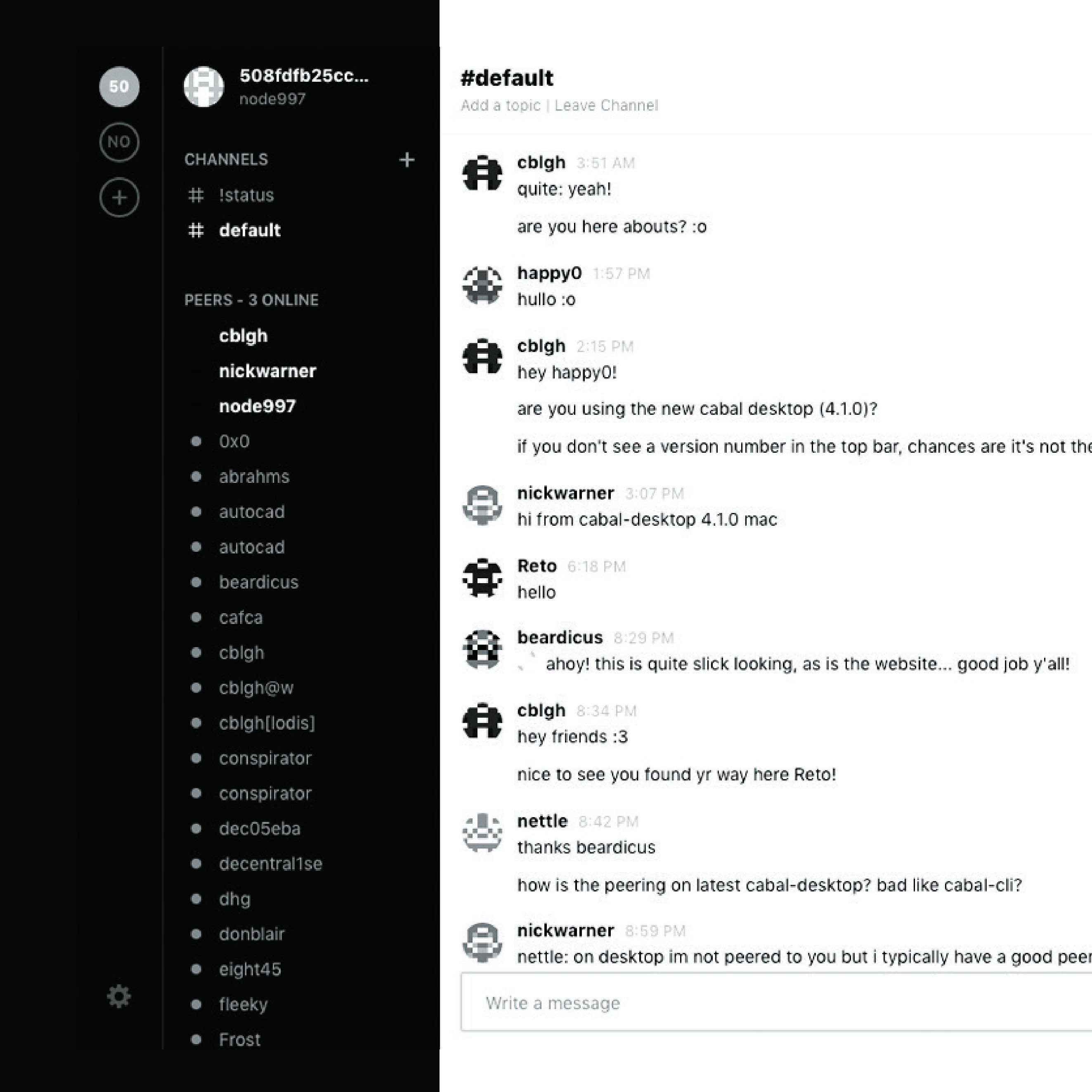
Using the app is ridiculously simple. First head over to `https://cabal.chat` and download the desktop app, which is available for Windows, Mac and Linux. There's also a command line version on there.

Launch Cabal, then simply copy/paste the public `cabal://` address to whichever group chat you want to join. After that, add your desired nickname and click 'Join'.

That's it! In the background, the app works similarly to Dat, and finds other peers connected to the same address. You'll be added to a standard looking chat room, where you can interact with the other users. You can add multiple channels like Slack, and configure different channel settings.

GOOD TO KNOW

While all chats users are encrypted, the IP addresses of those connected are not hidden. Also be aware that anyone with access to your `cabal://` address key can join your chat.


50
NO
508fdfb25cc...
node997
CHANNELS
!status
default
PEERS - 3 ONLINE
cblgh
nickwarner
node997
0x0
abrahms
autocad
autocad
beardicus
cafca
cblgh
cblgh@w
cblgh[lodis]
conspirator
conspirator
dec05eba
decentral1se
dhg
donblair
eight45
fleeky
Frost
#default
Add a topic | Leave Channel
cblgh 3:51 AM
quite: yeah!
are you here abouts? :o
happy0 1:57 PM
hullo :o
cblgh 2:15 PM
hey happy0!
are you using the new cabal desktop (4.1.0)?
if you don't see a version number in the top bar, chances are it's not the
nickwarner 3:07 PM
hi from cabal-desktop 4.1.0 mac
Reto 6:18 PM
hello
beardicus 8:29 PM
ahoy! this is quite slick looking, as is the website... good job y'all!
cblgh 8:34 PM
hey friends :3
nice to see you found yr way here Reto!
nettle 8:42 PM
thanks beardicus
how is the peering on latest cabal-desktop? bad like cabal-cli?
nickwarner 8:59 PM
nettle: on desktop im not peered to you but i typically have a good peer
Write a message

>COMMAND LINE CHEATSHEET_

This guide will show you how to carry out a bunch of cypherpunk-related tasks directly in the Linux terminal. These should work with most flavors of Linux, and some even work on Unix-like system such as Mac OS.

(N.B. Some commands that appear on more than one line, should all be typed on the same line. This is due to the formatting of the zine)

GENERAL ENCRYPTION

The simplest way to encrypt/decrypt files is with the OpenSSL utility. This is included with pretty much all Linux operating systems, so nothing needs to be installed.

ENCRYPTING FILES

```
$ openssl aes-256-cbc -in
~/File.jpg -out ~/Encrypted.file
```

"`aes-256-cbc`" This is the encryption cipher we'll be using for this example. To check out the different ones available, type in the `man openssl` command and read the manual.

"`-in ~/File.jpg`" This tells openssl that the input file (the one you want to encrypt) is 'File.jpg'. If you don't want to type out the full path for the file, you could just type `-in` (with a space after) and drag the file in the terminal.

"`-out ~/Encrypted.file`" This is the output file you will get. You can name it pretty much anything you like, although it's worth noting that you may run into problems if you name the output file the same as the input.

After you press return, you'll be asked to choose an encryption password. Type it in, and after you press return, you'll be asked once again to verify it.

Now if you check your home directory, you should have a new `Encrypted.file` file sitting there. This is an encrypted version of your original `File.jpg`.

You can now delete the original file if you want, using the `rm` utility. If it's particularly sensitive the `srm` command also works the same and is more secure.

```
$ srm  File.jpg
```

DECRYPTING FILES

In a terminal window, type the following and press return:

```
$ openssl aes-256-cbc -d -in
~/Encrypted.file -out ~/File.jpg
```

This is almost identical to the command for encrypting, except for a few things:

"`-d`" is included which tells `openssl` you want to decrypt the file.

The `-in` file is now the `Encrypted.File` and the `-out` is the original `File.jpg`.

Enter your decryption password. Press return and your original file will reappear in your home directory.

METADATA

We'll be using a utility called Mat, which stands for Metadata Anonymisation Toolkit. (This will only work on Debian-based Linux systems). Install Mat using this command, press enter.

```
$ sudo apt-get install mat
```

VIEWING METADATA

If you want to check the metadata on a single file, simply type the following and press enter, with `Example_File.jpg` being the file you want to look at.

```
$ mat -d Example_File.jpg
```

You can also check the contents of an entire directory. The `-c` flag will tell you whether each file in a directory is clean or not. Change the path to the directory you want to check.

```
$ mat -c Example_Folder
```

REMOVING METADATA

If you want to clean a single file, type the following and press return. You'll see a confirmation if all goes well.

```
$ mat -d Example_File.jpg
```

To clean the contents of an entire directory, you simply specify it's name instead of a file, then press enter. Again, you will see confirmation if successful.

```
$ mat Example_Folder
```

CHECKSUMS

Software devs often post file checksums on download pages and/or social media as a way for users to verify that they've downloaded the exact file as the dev intended. If the file has been tampered with or corrupted in transit, the output hash will be different.

VIEWING FILE CHECKSUMS

We can use the `shasum` utility to perform this task. Type the following, replacing the filename with your own, then press enter.

```
$ shasum -a 256 Example_File.bin
```

Note that the `-a 256` option allows you to choose which algorithm you want, in this case SHA 256. There are others availables, and you can view them in the man page by running `man shasum`.

Output will be a long alphanumeric string, e.g. `028901a417f96b286b309e48e2700a5d2fdbc0f29e5b34538511db3170cf9787`, and this is unique to your file.

This is what you compare to the string posted by the developer. If it's different, the file may be corrupted, or has been tampered with.

GPG FILE SIGNATURES

Another way to verify the integrity of files is with public key cryptography and GPG.

You will first need the signed software package that you wish to verify (in this case, I called it `example_file.tar.bz2`).

You also need the corresponding signature, which basically has the same file name, with the `.sig` suffix on the end.

Continued ⟶

Thirdly, you need the public key from whoever signed the package in an ASCII text file. In this example, the file is called `public.asc`. Most software teams post this information on their websites or publicly on social media.

VERIFYING SIGNATURES

If you haven't already, import the public key by typing the following, then press enter:

```
$ gpg --import public.asc
```

If all is well, you'll see a confirmation

Now we need to verify the fingerprint of the public key. These are usually posted on either the developer's website, or on social media.

The aim is to match what you see on your terminal with what has been publicly posted in order to prove it's from the correct person.

```
$ gpg --fingerprint 4F25E3B6
```

The final step is to verify the software package by inputting the following, and pressing enter:

```
$ gpg --verify
example_file.tar.bz2{.sig,}
```

The bit you're looking for is "Good signature" message. The key ID should be the same as the one you imported a few steps back.

ENCRYPTED EMAIL

GNU Privacy Guard or GPG is an application that facilities public key cryptography, and is widely used for sending and receiving encrypted emails.

For the beginners out there, each public key has a corresponding private key, which only the owner of the key has access to, so the idea is that messages can be cryptographically signed with a person's private key in order to prove it comes from the correct person.

It’s the same idea as verifying signatures in the previous section, but used for emailing.

People share their public keys on their sites, social media, or websites like Keybase, as a way to publicly tie this identity to themselves.

GENERATING A NEW KEYPAIR

If you don't have you're own keypair, type the following, and press enter:

```
$ gpg --gen-key
```

Follow the prompts to generate your key.

EXPORT PUBLIC KEY

```
$ gpg --export -a "User Name" > public.asc
```

Change `"User Name"` to whichever name is associated with the key you wish to export. This will create the file `public.asc`, which you can share with others.

DELETE PUBLIC KEY

If you want to remove a public key from your public key ring, enter the following:

```
$ gpg --delete-key "User Name"
```

ENCRYPT EMAIL

```
$ gpg -e -u "Sender User Name" -r "Receiver User Name" myfile.txt
```

Change the `"Sender User Name"` and `"Receiver User Name"` to the names associated with the keys in your key ring, and obviously change the filename to the email message you wish to send.

You'll be asked for your passphrase, and a new file `myfile.txt.gpg` will be generated. This is what you add as an attachment in your email/web client. Be aware the original file still exists too, so if you need to, delete it.

DECRYPT EMAIL

```
$ gpg -d myfile.txt.gpg
```

Input the above, then press enter. You'll be asked for your passphrase, then after it's successfully entered, your newly readable email file will appear in your home directory.

Again, remember the original file will still exist.

THE ZERO TERMINAL PROTOTYPE (V3)

For those who haven't seen the previous versions of the Zero Terminal, it is a Raspberry Pi Zero-W all-in-one computer, primarily designed to be used through the text console (hence the name).

The command line is a ridiculously powerful tool in the right hands, and I have always wanted to make my own system that's worthy of being THE go-to for portable Linux activities. This is why in the years since the last design, I have been thinking hard about how to go about it.

What I've come up with is a prototype which takes a different approach than before, this time aiming for a highly modular platform which gives the user lots of expansion options.

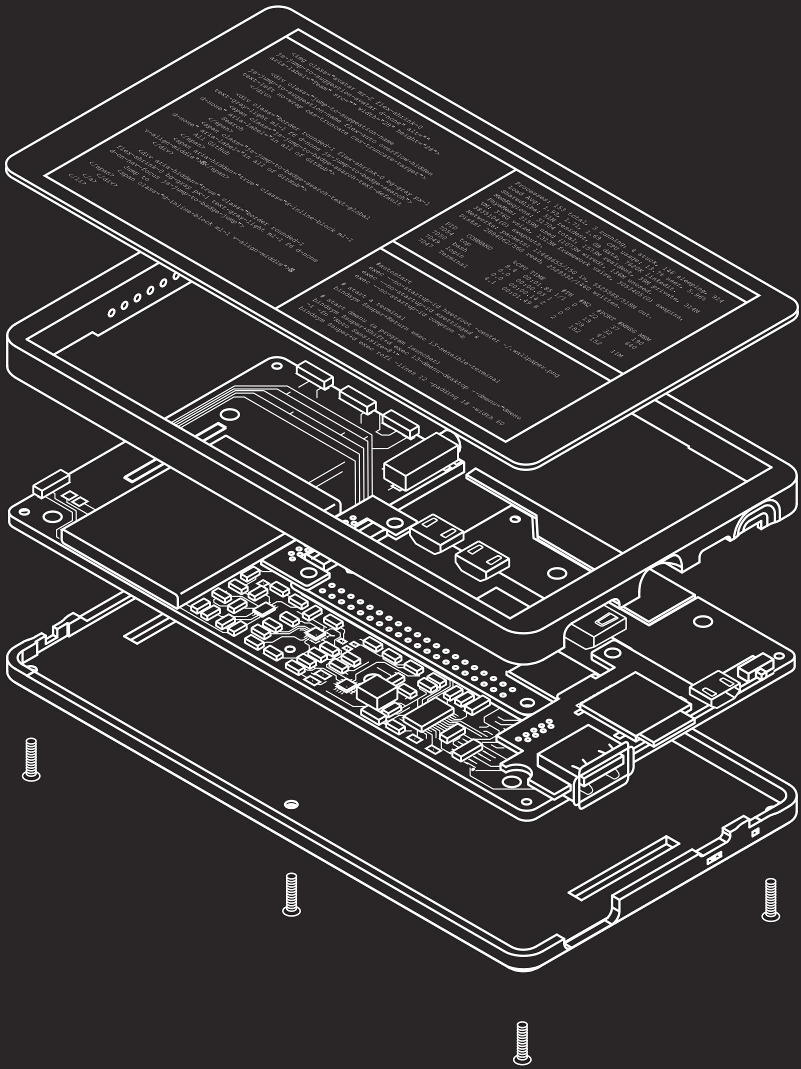
The design basically revolves around the Waveshare AMOLED 5.5inch 1080p touchscreen, which is a great piece of tech.

The entire device ends up being about the size of a fat smartphone, at about 13mm deep.

On the back there are two PCB sockets, and this is where the real power of the Zero Terminal becomes apparent. These are 2x 40pin sockets which connect directly to the Pi's GPIO. The idea is for this to be a platform where others can design a range of *back-packs* which plug directly into the back of the device to increase it's abilities.

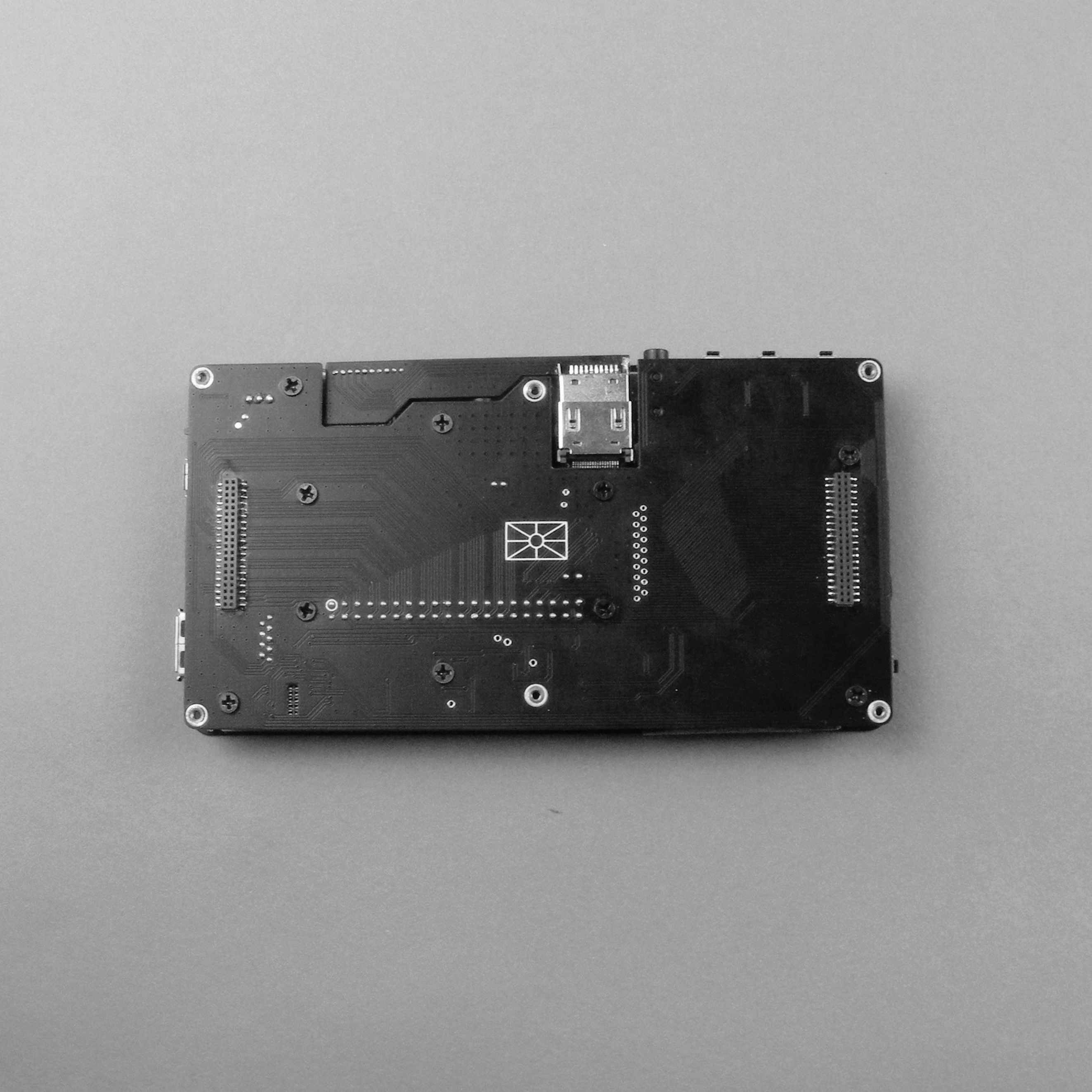
The first prototype backpack I have created is a slide-out keyboard. Combine that with i3 window manager, and you have quite the productive handheld Linux machine. Even though the Zero isn't the most powerful computer, you can still get a lot done through the terminal since it uses up a fraction of the resources that a GUI does. It still needs quite a bit of work, but you can see the potential for something like this.

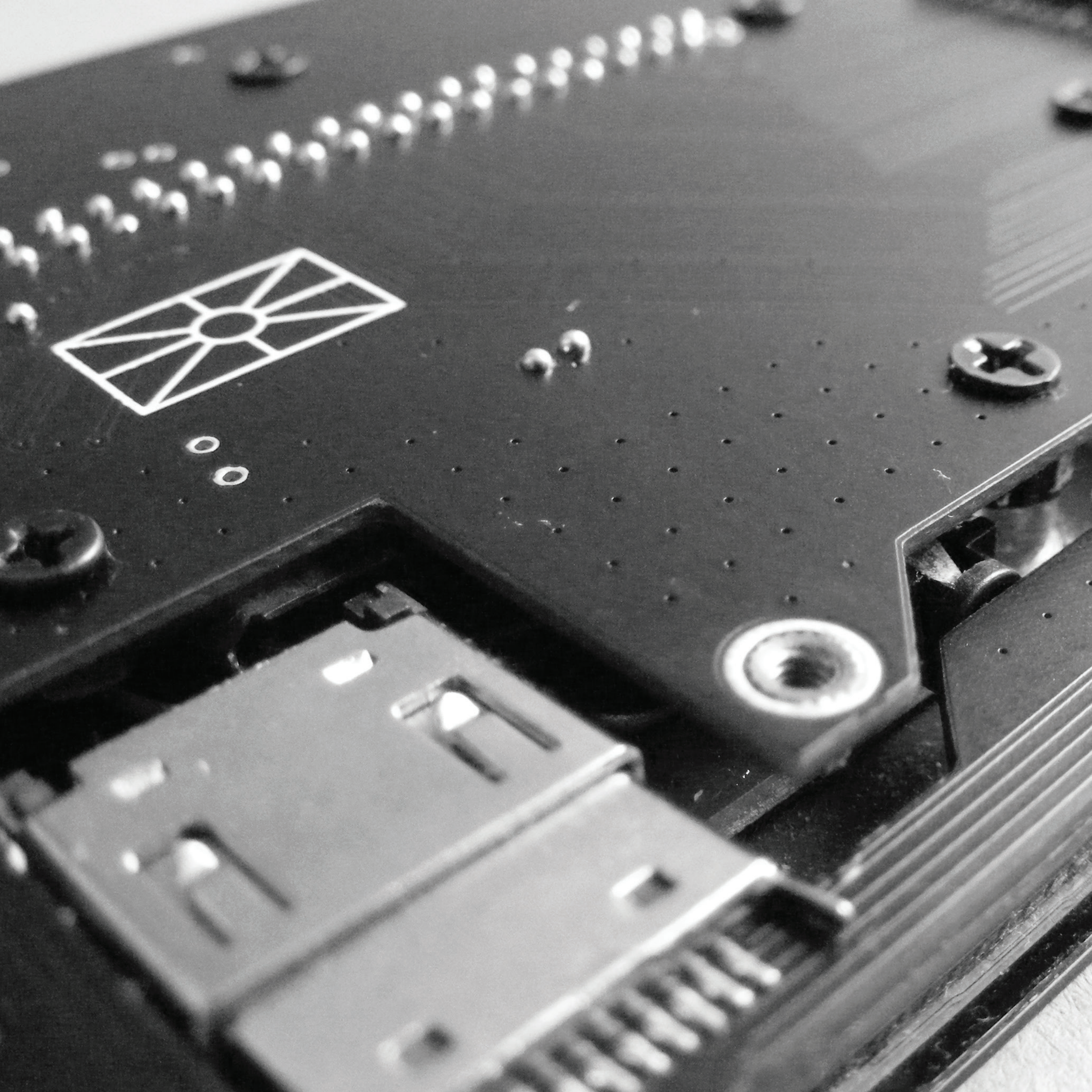
I can imagine a mixture of application specific backpacks being made to suit peoples needs. Things like radio transceivers, extra network interfaces, drone controllers, TV tuners, solar panels, and simple stands are all easily doable.




esc
Y
G
H
C
V
B
N








Esc
On
Off
Del
Enter
Tab
Caps
Shift
Shift
Ctrl
Fn
Menu
Dial
Backlight
Help
PgUp
Home
PgDn
End

The 1080p touchscreen also lends itself to custom programs that would help increase the usability of such applications. I even experimented with the "matchbox-keyboard" utility, which adds an on-screen Android-like keyboard to the touchscreen.

I have registered `ZeroTerminal.org`, which is currently redirecting to `N-O-D-E.net`. Over the coming months, I want to make a website to help build up the platform, showing people exactly how to make these, and showcasing all the backpacks and custom apps other users create. Eventually it would be great to even design our own screen boards, allowing us to make everything even smaller.

COMMAND LINE APPS

Here are a few great text based apps that work well on the Zero Terminal (and Raspberry Pis in general).

Firstly, if you've ever wondered how to install **i3** on Raspbian, this guide will show you how:
`steemit.com/@joedoe47/easily-run-i3-on-raspberrypi`

Micro is an excellent text editor, that looks and works very similar to other editors like Sublime Text. Highly recommended.
`https://micro-editor.github.io`

Midnight Commander is a visual file manager that works through the terminal.
`https://midnight-commander.org`

Htop, the interactive process viewer. This is a must for keeping an eye on the system.
`https://hisham.hm/htop`

Browsh is a modern browser that works entirely inside the terminal. Images are replaced with ASCII to significantly reduce bandwidth and browsing speed.
`https://brow.sh`

Newsbeuter is a cool little utility for managing and viewing RSS feeds.
`https://newsbeuter.org`

AKASHA INTERVIEW - DECENTRALIZED SOCIAL NETWORKING WITH MIHAI ALISIE

One of the crucial components of the emerging decentralized web is the need for a good P2P social network. AKASHA, which stands for 'Advanced Knowledge Architecture for Social Human Advocacy' is one such contender taking on the challenge.

Founded in 2015 by *Ethereum* and *Bitcoin Magazine* co-founder Mihai Alisie, the project uses both the Ethereum blockchain and IPFS to deliver a social network similar to Medium or Reddit—all without any central servers.

I spoke with Mihai about the project, his thoughts on free expression, and where AKASHA could go in the future.

Check it out →

Why was AKASHA started?

Our journey started as an experiment to see if it is possible to create a free channel of expression working in a completely decentralized fashion.

In order to achieve this we used things like Ethereum smart contracts and IPFS wrapped up in a simple-to-use social application.

Since then we've been compared many times to today's social media platforms but our intention was never to become "the new Facebook," "the new Twitter," or the new "X."

Our intention was, and remains, to create something simple to use but meaningful in the way it works—proving in the process that "there's a better way now" to tackle BIG problems, even if sometimes they seem almost "impossible to solve."

This is how we started thinking about blockchain technology in the context of our freedom of expression as individuals, since individual self-expression is critical to maintaining a healthy balance in societies.


My first board
Latest entries
Lates
0xb47b...7de9
0 min read | Published 20 days ago
Новый блог об обществе, крипте, линуксе и т.д.
Всем привет. Я начал новый блог. Он пока ещё довольно уныл, но это поправимо. Подписываемся, предлагаем свои свои темы.
RUSSIAN
BLOG
GAB
SOCIETY
LINUX
CRYPTO
0
0
The voting period has ended


#VERBIER
AKASHA


RØDE
MICROPHONES



But, in the phase that we announced on May 3rd 2019 under the name "AKASHA Reloaded" we outlined another side to the problem in the form of collective freedom of expression, often ignored and/or suppressed.

If you think about it, our ability to come together as one and express ourselves as a collective is one of the most powerful and natural thing we can do as social creatures but the tools enabling this type of interaction are either limited or inexistent.

We call these pockets of individual and collective freedom AKASHA worlds.

How long have you been working on this, and how big is the team?

The project started in July of 2015, with the first prototype being developed by Marius in Q4 of the same year; we closed out that year as a team of 4 people, driven by passion and inspired by "what-ifs."

On May 3rd, 2016 also known as World Press Freedom Day, we publicly announced the AKASHA Project. We chose this day as the cause of press freedom is something which is extremely important to the team and project as a whole. The suppression of the fundamental human right to freedom of expression anywhere causes damage to everyone everywhere, and in these times of government overreach and corporate control, it has become more and more of a day to reflect upon the problems that we need to solve as a society.

We sent alpha 0.1 invitations in Q4 of 2016, with the first public alpha, 0.2, being released on January 16th of 2017. By the end of that year AKASHA had grown to a team of 6 wonderful people. We went on to release our public beta in February of 2017, calling out for people to help us stress test the app, smart contracts, and crypto-economic assumptions. By the end of 2017 AKASHA comprised 7 people, and we were on the lookout for more people to join us on our journey!

In 2018 we engaged in some deep research to find solutions to some of the problems that had arisen after the beta release, put out a web version of the beta, presented at the Heidelberg Laureate Forum, and grew to 12.

Last year this whole process of researching, questioning, and conceptualizing culminated in the posting of two long-form posts: Metamorphosis I and II, which you can read on our blog (`akasha.org/blog`). Both of these posts explored the ideas that had brought us to this point of our journey, and gave some clues as to the direction that we were intending to take the project in through 2019, 2020, and beyond! On top of this, we also presented at the World Economic Forum, joined the World Wide Web Consortium, and celebrated World Press Freedom Day once again by announcing the AKASHA World Framework through the post "AKASHA Reloaded: Three Spins Around The Sun Later"!

We followed up this announcement at Devcon 5 in Osaka by unveiling `ethereum.world`, the first world to be built using the framework. By the end of 2019, the team stood at 15 people, and looking to the future we are sure it won't stay that way for long.

One of the big challenges of P2P systems is decentralized moderation. What kind of system does AKASHA use to limit things like spam and other unwanted content?

At the moment this is kind of an open question as we envisage future worlds needing different methods for moderating things. We want to empower world creators by giving them access to a variety of ways that they can curate the content that exists in the worlds that they bring into being, with all of these sharing the same open data layer powered by IPFS.

What platforms can you use AKASHA on? and how easy is it for users to manage their identities on various devices?

The first "world" built with the AKASHA World Framework will be `ethereum.world`, a project seeking to unify the Ethereum ecosystem by integrating decentralized apps into a simple, delightful social experience. Those interested are invited to sign up for early access, suggest features, propose integrations, and nominate values using the `canny.io` board hosted on the site.

Ethereum apps and services integrate once but can be reused everywhere, and so any steps taken to bring in projects at this early stage will have an effect upon what is

available to future users and world builders. The platform approaches identity as an integration, and so there will be a variety of options in terms of identity providers when it comes to accessing the platform; if any readers have services that they think it would be unwise to overlook, we'd appreciate them submitting them via `ethereum.world`.

Nice, so what are your long-term plans for the wider AKASHA project?

Looking at the bigger picture, the ecosystem approach will enable people using the platform to float between any following worlds built with the AKASHA framework, finding different communities, different pieces of functionality, and different causes.

We are working to provide people with a tool that can unlock the latent power of humanity, expanding our collective minds at local, regional, and global scales.

That seems like a huge and worthy goal.

Where's the best place to go to learn more, and perhaps contibute to the project?

People looking for further information should visit our website, especially the posts that have recently been written up by our dev team (`akasha.org/blog/category/dev/`), as well as visiting our YouTube channel and checking out our DEVCON 5 presentation (`youtu.be/CaMXa6R-jyI`).

Thanks Mihai.

All the best for you and the team. I'm very interested to see where the AKASHA project goes in the coming years.

THE NFC CARD

The NFC Card is a data card that lets you wirelessly transfer different types of data to phones and other NFC readers.

Much like the NFC Keys mentioned previously on the NODE website (`N-O-D-E.net`), these cards have multiple chips with corresponding buttons, so you can program and read different data on demand.

The button switches break the antenna circuit, so the chips can only be read when the buttons are pressed. This prevents passive reading which standard NFC tags, stickers and tap-to-pay cards are susceptible to.

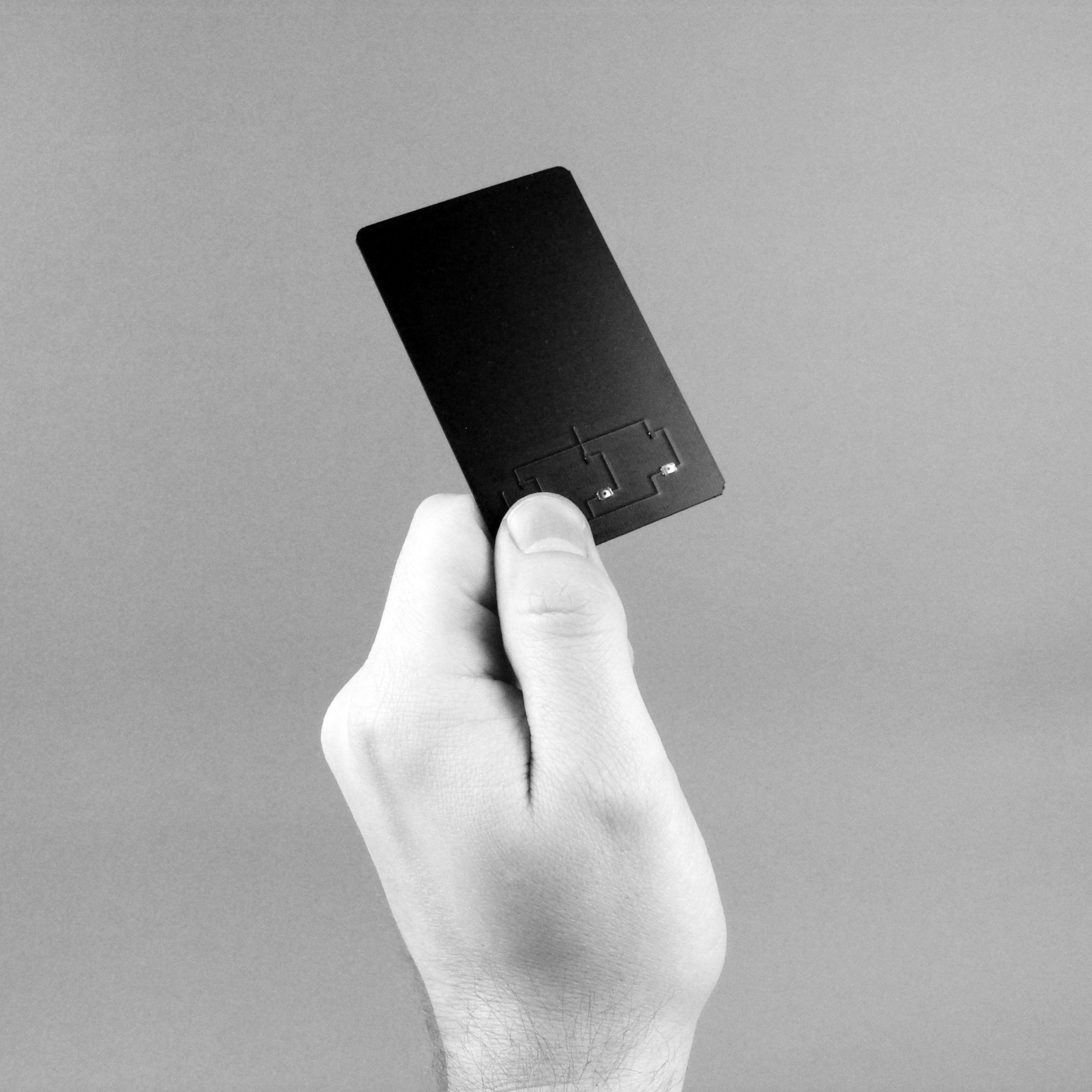
The design is credit-card sized (86 x 54mm), so it will fit inside your wallet with no problem. I also intentionally used extremely low profile components for the chip and buttons (less than 1mm height) as well, leaving us with a size and thickness that's basically the same as a regular credit/debit card with embossed numbers & characters.

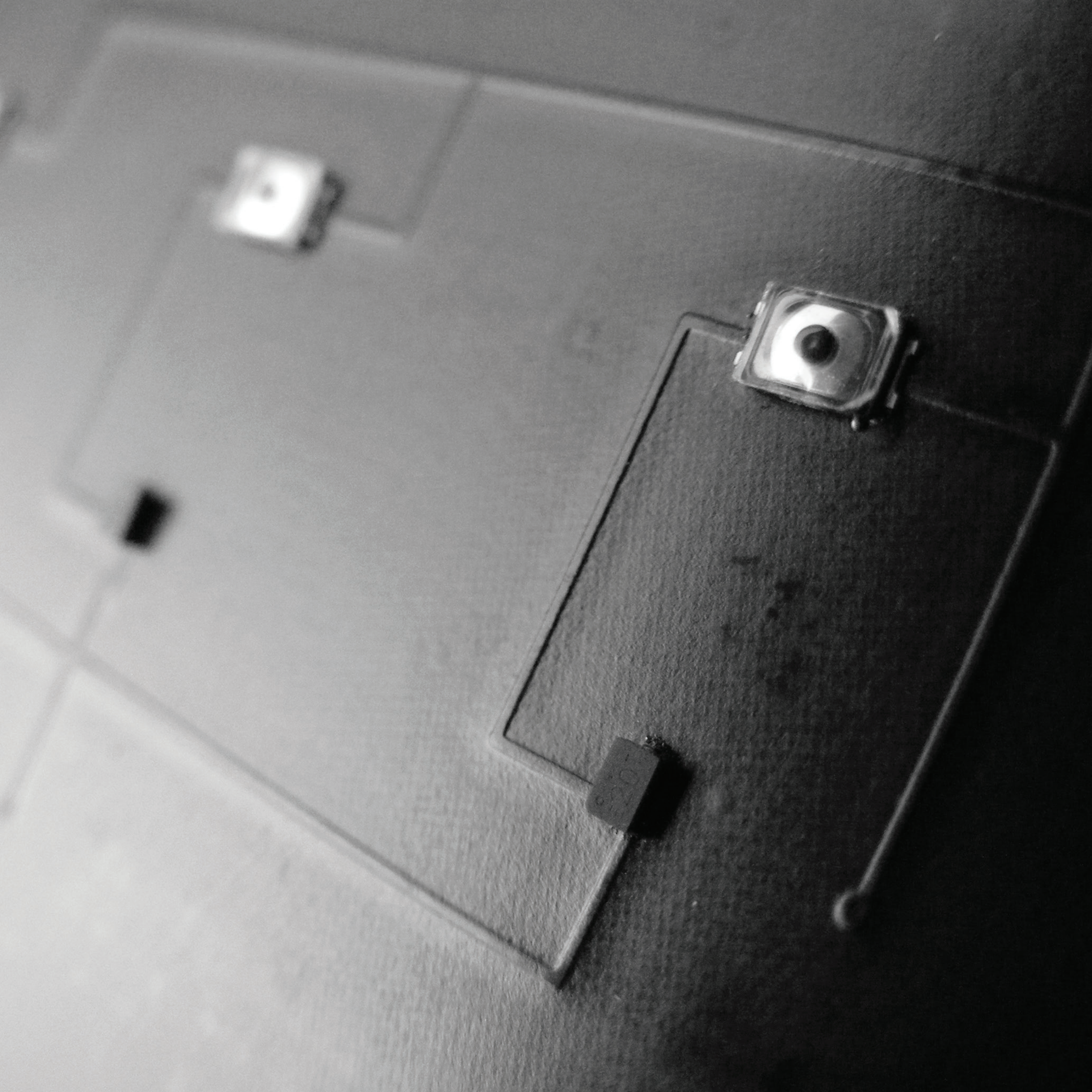
The tiny chips have 106 bytes of storage each, making it useful for various applications; crypto address, URLs to GPG keys, websites or social media links. Phone numbers would also work, as well as door access and other ID (Though it depends on the RFID system used).

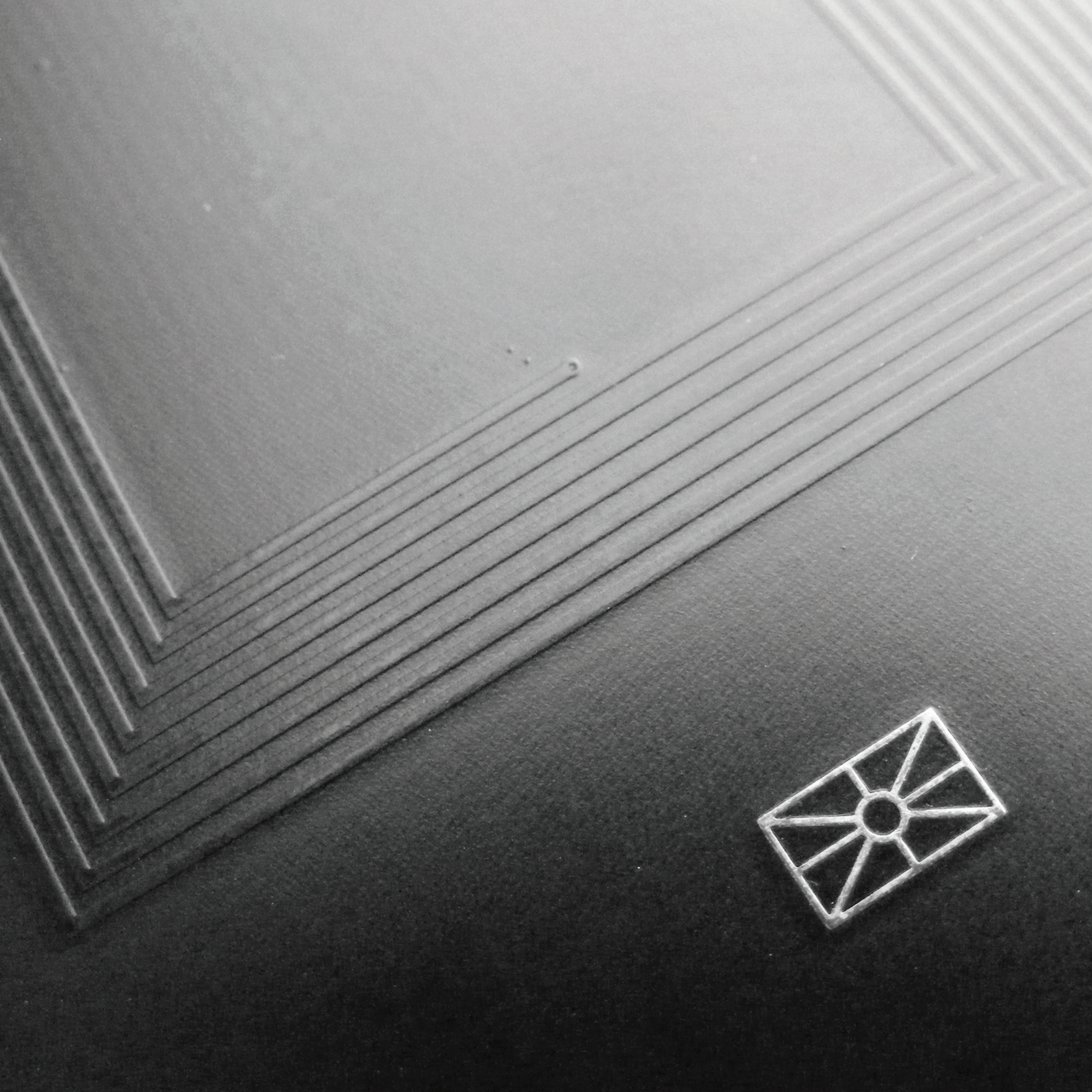
To make your own, the PCB specs are as follows: 86x54mm board, 0.8mm thick FR4 material, 2 layers. The files will be on the NODE site, so search for them.

The parts used in this design are the "SL2S2002FTB,115" NFC chip, and the "EVPAWCD4A" single pole, single throw tactile switch. Both are very small surface mount components, and will require reflowing.

If you're going to design one from scratch, use an antenna generator to estimate the size needed. Remember that you'll have to make sure the inductance from your antenna design generates the correct capacitance for the chip you use. Too low and the chip won't power on, and anything too high will permanently damage it. Most chips have a little wiggle room, so don't worry if it's slightly out. Read your chips datasheet for more information.







CONNECTING THE INTERNET - HOW BGP WORKS, AND SOMETIMES DOESN'T

In 1989 the Internet as we knew it was on the verge of collapse. This wasn't due to limited adoptability or lack of interest--quite the opposite--for the first time ever, individuals outside of academia and the military were going online in droves. The network was overloaded, and nobody had a solution.

The Internet at the time looked very different than it does today. The National Science Foundation Network (NSFNET), birthed in 1985, was created as something of a successor to the aging ARPANET project. Starting with five supercomputing centers linked via 56 kbit/s links, the network would expand to 13 nodes utilizing 1.5 Mbit/s (T-1) links by 1988. These backbone nodes allowed for connection to 10 regional networks which in-turn allowed access to 170 additional networks across the United States. This was an amazing advancement for the exchange of information online, but with the inclusion of all of these networks, the NSFNET as a whole would become more congested.

At the time, traffic was routed between networks within NSFNET using a relatively young protocol called Exterior Gateway Protocol (EGP). EGP, formally specified in 1984, allowed gateway hosts within disparate networks to talk to a core router that would act as a traffic controller for sending data in and out of the separate networks it linked together. EGP structured the network in a tree-like topology, where smaller networks would connect up to these core routers which would then be connected to other core routers within the larger NSFNET. While EGP worked well initially, as the network grew it started to show issues as routing data began to take up more network traffic and routers struggled to handle growing lists of routes that specified how data was to traverse the network.

The big fix for NSFNET's routing problem came in 1989 from two friends, Kirk Lougheed of Cisco Systems and Yakov Rekhter of IBM,

having lunch at an IETF meeting in Austin, Texas. The pair recorded their design for a new routing protocol onto two napkins, which would be developed into what is now known as Border Gateway Protocol (BGP). Formalized in June 1989, Border Gateway Protocol is often referred to as a "two-napkin protocol" to showcase its simplicity. At the time, BGP was considered something of a hack or quick fix to the the existing routing issues--there wasn't much thought put into security, or future-proofing the design. Network operators, valuing working solutions and simple designs over bureaucracy, quickly adopted the protocol. BGP became the de facto way of routing traffic between networks in short order.

The fundamental change with the introduction of Border Gateway Protocol was the shift to a mesh topology. While Exterior Gateway Protocol required networks to connect together via core routers, BGP allows networks to connect directly with one another. Through mesh, networks could connect in a decentralized fashion, with no reliance on centralized network points. Additionally, BGP improved traffic flow by implementing a "best path" algorithm (known as path vector routing) allowing routers to advertise which routes they have access to for facilitating traffic flow within the NSFNET. Essentially, each router can show the paths by which data can flow through them and to a destination. Further, network operators received the ability to be selective with where their traffic went; if an operator doesn't want their traffic to go through a competitor or a known unstable network, they could choose to route around it.

To fully understand how BGP works, we have to introduce the concept of an Autonomous System (AS), which is essentially just a single network within a larger internet. When traffic flows across an internet, it travels through various autonomous systems before reaching a final autonomous system containing the destination machine. There can be many routes to get from one autonomous system to another (traveling through other, different autonomous systems along the way), but most BGP routers are configured to prefer the shortest path (least number of hops, networks to pass data through) to get data to a destination the fastest.

GETTING ON THE INTERNET

But how do networks even get on the Internet in the first place? It all starts with getting IP address allocations. This is normally done via an IANA Regional Internet Registry (RIR) such as AFRINIC, ARIN, APNIC, LACNIC, or RIPE NCC depending on where the autonomous system is physically located. If using IPv4, a network needs a minimal allocation of 256 addresses (referred to as a /24, with a range like xxx.xxx.xxx.0-xxx.xxx.xxx.255) to be routable on the IPv4 Internet. However, due to registries running out of IPv4 addresses to allocate, organizations must buy address blocks second-hand from private organizations. If using IPv6, a network needs a minimal allocation of 1,208,925,819,614,629,174,706,176 addresses (commonly referred to as a /48, with a range like xxxx:xxxx:xxxx:0000:0000:0000:0000:0000-xxxx:xxxx:xxxx:ffff:ffff:ffff:ffff:ffff) to be routable on the IPv6 Internet.

Aside from addresses, an organization also needs to register for an Autonomous System Number (ASN) to uniquely identify the network on the greater Internet. ASNs are formatted with the letters "AS" followed by a string of digits, to create something like "AS1234."

Currently, IPv4 addresses cost $25 each on the second-hand market, meaning an AS would need to spend a minimum of $6,425 if they wanted to use IPv4 addresses. IPv6 addresses are significantly less expensive, coming out to fractions of a penny for each address. For the minimum /48 allocation, an AS would have to spend only $250 if they wanted to use IPv6 addresses. Acquiring an ASN also comes with an associated cost of $550. If an AS wanted to proceed with registering an ASN and acquiring both IPv4 and IPv6 address space, the total upfront costs would be $7,225. Additionally, there is an annual maintenance fee of $350 (waived at time of initial registration, going into effect the following year).

THE PHYSICAL INTERNET

After an Autonomous System has all of their designations, they still have to physically get on the Internet. This is often done at Internet eXchange Points (IXPs) and various Carrier

Hotels (data centers where many Internet Service Providers have infrastructure). These locations are usually housed in large, robust buildings that are often fireproof and sometimes even bombproof.

As they are critical to keeping the Internet operating, these facilities may also be guarded and appear unassuming from the street. Within them, many networks have a physical "edge" of their network known as a Point of Presence (PoP). A PoP typically consists of some amount of servers and networking equipment, and is commonly used for connecting a network up to Internet Service Providers (ISPs) and other networks.

Being located in one or more IXPs allows different organizations to more freely and easily share data with other, neighboring networks. Keep in mind however, that not all networks need to have a physical presence in a data center to become a part of the Internet. Many major cities have regional fiber networks spread underneath the streets that can connect buildings up to an ISP. A provider might hand off a cable with a BGP session to their router sitting at the other end--just as though you were physically connected to them in a data center!

Much like with a home LAN, data centers will often run layer 2 networks that allow two networks to connect their BGP routers with one another for traffic exchange. Some data centers may even simply run an ethernet cable directly between two BGP routers to facilitate a connection.

HOW NETWORKS CONNECT

While it is obvious by now that all organizations on the Internet talk to one another via BGP, there are different relationships between networks on the Internet that dictate how networks can connect to one another.

Network connections can be broken down to two groups: peering and upstream transit. When two networks want to connect with one another, they must negotiate a connection deal. Oftentimes, a network will have one or more upstream providers that can route traffic upward to the larger Internet as a whole,

much like you may see with a residential ISP and one of their customers. This is almost always a relationship where a customer is paying a provider.

In contrast, peering can exist between two networks of somewhat equal size and position where it is deemed mutually beneficial to connect. For example, an organization like Google might want to peer with an organization like Digital Ocean so users on each network can access resources on the other directly, without having traffic traverse networks owned by upstream transit providers. Users on both networks get the benefit of having fast access that is just one hop away, while the networks themselves can benefit from this direct connection by bypassing any upstream providers who would normally charge transit costs. Peering is often done freely with no money exchanging hands as both parties benefit from it equally.

When networks actually connect with one another, they are essentially exchanging route information. Each network advertises IP ranges that they control, while also potentially sharing routes for other networks that can be reached through them. All networks will have IP addresses that they advertise as being available to access, but not all of them will allow others to route traffic through them to more distant networks.

THE TIERED INTERNET

When networks connect to form a mesh, there is still some semblance of a hierarchy to the different networks based on size and ability. The Internet as we know if is split into 3 tiers, forming something of a pyramid.

Tier 1 networks (at the top) make up the backbone of the Internet, and usually have physical infrastructure spanning a whole country. These are large networks like AT&T, Sprint, Verizon, Century Link (and formerly Level3), etc. Below the tier 1 networks are the tier 2 networks. These are still mostly large ISPs, but are more regional and need to purchase some transit from a larger tier 1 network to get a whole view of the Internet. Some examples of tier 2 networks include Cogent, Comcast, and Hurricane Electric.

At the bottom of the pyramid, there are tier 3 networks. In general, tier 3 networks are "everything else," and can include smaller last-mile ISPs, businesses, and schools. Tier 3 networks will always need to purchase transit from a tier 1 or tier 2 network for access to the entire Internet. Considering relationships based on this tier system, smaller networks will be paying larger networks for transit as you work up the tiers. However, networks that exist on the same tier will often peer with one another without paying any money for that connection. For example, if Comcast uses AT&T for an upstream provider, Comcast will pay AT&T for transit, but Comcast and Hurricane Electric may exchange traffic freely between one another for no payment as it benefits both of their networks (similar to the example with Google and Digital Ocean explained earlier).

An interesting phenomenon we are now seeing within the Internet is something called "Donut Routing," where larger, tier 1 networks are often being routed around to avoid transit costs. As smaller networks gather more peering agreements, they can reduce their overall costs and rely less on larger networks.

BGP SECURITY

When BGP was written, there were very few security precautions taken for assuring proper use of the protocol. Even now, decades later, few network operators implement anything for security purposes. Initially, there was no thought that anyone would do anything nefarious with the BGP protocol, and even today, most operators rely on the honor system to make sure their peers are behaving properly. That said, BGP security incidents happen almost every day! For every issue you may hear about in the news, there are dozens that are never heard about.

There are three main types of BGP security incidents that can occur: a route leak, route hijacking, and denial-of-service (DoS) attacks. A Route leak occurs when content in the BGP routing table is accidentally or maliciously altered, so traffic can't reach its intended destination. This is like a network is saying "use me to get to this destination" when it cannot properly route traffic or wants to eavesdrop. Route hijacking is when a bad actor announces a victim's IP addresses, rerouting all target traffic to itself. Denial-of-service

attacks occur when a bad actor sends undesirable BGP traffic to a victim, exhausting the victim's resources (knocking them offline).

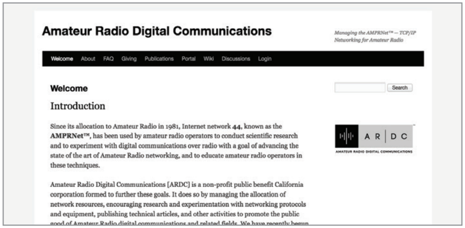
To combat these various security concerns, some techniques have been introduced via RFCs to make BGP less of an attack vector. The biggest contribution so far is something called Route Origin Validation (ROV) which uses public key infrastructure to make sure routes are signed by the Autonomous System that is originating them.

Further, this same technology is used by the BGPSec standard to have each router in a given path use signatures to verify the authenticity of a given hop to create a fully trusted chain from origin to destination.

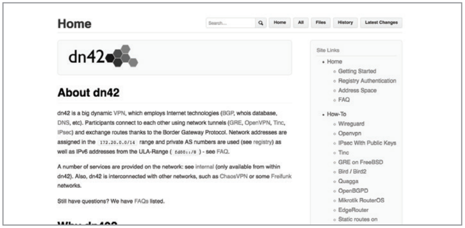
These standards are currently optional, and as of October 2019, only 84 Autonomous Systems out of a pool of over 92,000 currently use Route Origin Validation. Until more networks adopt ROV and/or more security practices are introduced, we will continue to see incidents of traffic being hijacked or spied upon, often without users even knowing that it is happening.

EXPERIMENTING WITH BGP

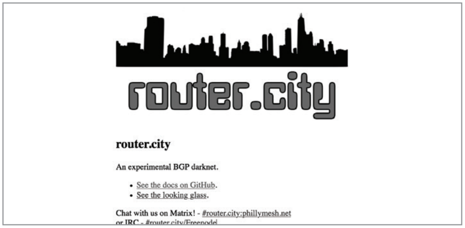
As we have shown, it is normally quite cost-prohibitive to get on the Internet yourself and operate a BGP router. However, there are still ways you can get on the Internet with BGP, or generally experiment in a safe setting.



If you happen to be a licensed amateur radio operator, you can get a /24 allocation (256 addresses) of IPv4 from AMPRNet, an experimental network for ham radio operators. These are real, routable IP addresses and you can configure them for use via several methods from AMPRNet themselves or you can ask a hosting provider to announce these addresses for you so they are available on a virtual private server under your control. More information is available at `https://www.ampr.org`



If real-world BGP routing seems a little daunting, you can join the amateur DN42 network. DN42 is a BGP test network where everyone is given IP addresses and AS numbers out of private ranges. Users are encouraged to establish BGP sessions with one another to learn how BGP works. More information is available at `https://dn42.eu`



Alternatively, I have been involved with developing a BGP test network framework called router.city. Like DN42, this network uses addresses and ASNs out of private ranges, but is something easy to clone and set up (with example configuration documents) between a small group of people who want to play with BGP in a setting where they can create disruptions and do more experimental work. More information is available at `https://router.city`

CONCLUSION

Though the Internet can be a difficult entity to see and understand, it is really a fairly basic network connecting smaller networks through a common language.

While Internet routing knowledge is usually reserved for a select group of network engineers, it is important that information about how it is set up and operated is known to more people.

How to connect to the Internet shouldn't be a secret held by large organizations, and hopefully this inspires you to get more involved with getting online!

OPEN SOURCE DIRECTORY

P2P / DATA

Airlock	*IPFS/Ethereum based file storage/sharing*	`https://github.com/slothbag/Airlock`
BitDust	*Decentralized online storage network*	`https://bitdust.io`
ClearSkies	*Dropbox-like file sync program*	`https://github.com/jewel/clearskies`
Dat Project	*File storage/syncing protocol*	`https://datproject.org`
GUN	*Decentralized graph protocol to sync the web*	`https://gun.eco`
Hola	*P2P virtual private network*	`https://hola.org`
IPFS	*P2P data distribution and storage*	`https://ipfs.io`
Lethean	*dVPN with bandwidth marketplace*	`https://lethean.io`
Lokinet	*Monero based decentralized VPN*	`https://loki.network`
Mysterium	*dVPN built on Ethereum*	`https://mysterium.network`
Nym	*Anonymous distributed VPN*	`https://nymtech.net`
OnionShare	*TOR based tool for anonymous file sharing*	`https://onionshare.org`
Orchid	*Ethereum-based VPN*	`https://orchid.com`
P2PVPS	*Decentralized VPS network*	`https://p2pvps.org`
Privatix	*Another Ethereum dVPN project*	`https://privatix.io`
Radicle	*P2P stack for code collaboration*	`http://radicle.xyz`
Scuttlebot	*Peer-to-peer log store*	`http://scuttlebot.io`
Sentinel	*dVPN with "Proof of Traffic" systemt*	`https://sentinel.co`
Storj	*Decentralized cloud storage*	`https://storj.io`
Syncthing	*Torrent/Dropbox type file storage/distribution*	`https://syncthing.net`

Continued →

P2P / ENTERTAINMENT

Bittube	*IPFS based video platform*	`https://bit.tube`
Decentraland	*Decentralized virtual reality world*	`https://decentraland.org`
D.Tube	*Steemit/IPFS base video platform*	`https://d.tube`
Funk Whale	*P2P music/audio network*	`https://funkwhale.audio`
LBRY	*Community run digital marketplace*	`https://lbry.io`
Livepeer	*Ethereum based video platform*	`https://livepeer.org`

P2P / COMMUNICATIONS

Adamant	*Blockchain-based anonymous messenger*	`https://adamant.im`
Aether	*Decentralized pulic communities*	`https://getaether.net`
Afari	*Blockstack based social network*	`https://afari.io`
Akasha	*IPFS/Ethereum based social network*	`https://akasha.world`
BitMessage	*Encrypted P2P messaging app*	`https://bitmessage.org`
Briar	*Encrypted messaging and forums*	`https://briarproject.org`
Cabal	*Experimental P2P group chat app*	`https://cabal.chat`
Iris	*P2P social/chat platform*	`https://github.com/irislib/iris`
Matrix	*An open network for secure P2P communications*	`https://matrix.org`
Peerlinks	*Distributed secure IRC*	`https://peerlinks.io`
RetroShare	*Cross-platform, secure, P2P communications*	`http://retroshare.us`
Ricochet	*Encrypted instant messaging routed through TOR*	`https://ricochet.im`

P2P / COMMUNICATIONS

Secure Scuttlebutt	*Decentralized secure gossip platform*	https://scuttlebutt.nz
Sigle	*Create decentralized blogs*	https://sigle.io
Tinfoil Chat	*Onion-routed messaging system*	https://github.com/maqp/tfc
Tox	*P2P audio, video and text communications*	https://tox.chat
Unwalled Garden	*Dat-based social network*	github.com/beakerbrowser/unwalled.garden
Whatsat	*Encrypted, onion-routed P2P chat*	https://github.com/joostjager/whatsat

P2P / NEW INTERNET

Beaker Browser	*Browser for Dat based decentralized web*	https://beakerbrowser.com
Blockstack	*Platform for building decentralized apps*	https://blockstack.org
Enigma	*Privacy layer for the decentralized web*	https://enigma.co
Ethereum	*P2P network for smart contracts*	https://ethereum.org
Handshake	*Decentralized naming protocol*	https://handshake.org
Maidsafe	*Creating a new alternative internet*	https://maidsafe.net
NameCoin	*Blockchain based naming system*	https://namecoin.org
Open Index Protocol	*Indexing the worlds data*	https://openindexprotocol.com
Solid Project	*Project to create a new decentralized web*	https://solid.mit.edu
SkyWire	*Decentralized platform to build a new internet*	https://skycoin.net/skywire
Swarm	*Serverless hosting, and P2P data distribution*	https://swarm.ethereum.org
YaCy	*Decentralized, user-run search engine*	http://yacy.net/en

P2P / NEW INTERNET

Zeronet	*Open, free and uncensorable websites*	https://zeronet.io

P2P / MONEY & COMMERCE

Bisq	*Decentralized cryptocurrency exchange*	https://bisq.network
Bitcoin	*A peer-to-peer electronic cash system*	https://bitcoin.org
Lightning	*Scaling network for Bitcoin transactions*	https://lightning.network
Litecoin	*Peer-to-peer internet currency*	https://litecoin.org
Monero	*Privacy-focused P2P digital currency*	https://getmonero.org
Open Bazaar	*A decentralized ecommerce platform*	https://openbazaar.org
Origin	*Marketplaces on the blockchain*	https://originprotocol.com

P2P / NETWORKING

Althea Mesh	*Incentivized mesh networking*	https://althea.org
Cjdns	*E2E encrypted IPv6 mesh networking*	https://github.com/cjdelisle/cjdns
DN42	*Decentralized private networking*	https://dn42.net
Hyperboria	*P2P, encrypted private networking*	https://hyperboria.net
LibreRouter	*Project to design multi-radio routers*	https://librerouter.org
Locha Mesh	*Chat / send Bitcoin without internet*	https://locha.io

P2P / NETWORKING

Open Garden	*Decentralized wifi sharing platform*	https://opengarden.com
RightMesh	*Ad-hoc mobile mesh networking platform*	https://rightmesh.io
Yggdrasil	*E2E encrypted IPv6 mesh networking*	https://yggdrasil-network.github.io

HEALTH & SCIENCE

DIY Particle Detector	*Cheap, buildable detector by CERN*	github.com/ozel/DIY_particle_detector
F.Lab	*Open source bioscience machines*	https://f-labth.blogspot.com
Four Thieves Vinegar	*DIY epinephrine autoinjector + more*	https://fourthievesvinegar.org
Glia	*Low-cost open source medical devices*	https://glia.org
MobileCG	*Open source clinical grade ECG*	https://github.com/peterisza/mobilecg
OpenAPS	*Automated pancreas system*	https://openaps.org
OpenBioMedical	*3D printable biomedical devices.*	http://openbiomedical.org
OpenFlexure	*High precision RPi microscope*	https://openflexure.org
OpenPCR	*DNA molecule copying machine*	https://openpcr.org
Open Detector	*Pocket-sized ion chamber*	https://hackaday.io/project/27508
Open RAMAN	*Low Cost Raman Spectrometer*	http://open-raman.org
Open Source Imaging	*A range of projects inc MRI machines*	https://opensourceimaging.org
PocketPCR	*Mini, buildable thermocycler*	http://gaudi.ch/PocketPCR
Tympan	*Hearing aid development platform*	https://tympan.org
Un0rick	*Open source ultrasound project*	http://un0rick.cc

AUGMENTATIONS

Alice	*Child focused robotic exoskeleton*	`https://indi.global/alice`
Fable	*Electronic prosthetic hand*	`github.com/openbiomedical/OBM-FABLE`
InMoov Hand	*Robotic hand build guide*	`http://inmoov.fr/hand-and-forarm`
Enabling the Future	*3D printed prosthetics*	`http://enablingthefuture.org`
OpenBCI	*Open source brain computer interface*	`https://openbci.com`
Open Source Leg	*Fully articulating robotic leg*	`https://opensourceleg.com`
StarCat	*Open source bio signals*	`https://starcat.io`
Neuroon	*Open sleep/neuro tracker*	`https://github.com/inteliclinic`

SPACE

Blockstream Satellite	*Satellite network based bitcoin transactions*	`https://blockstream.com/satellite`
CubeSat	*Open standard for DIY satellites*	`https://cubesat.org`
Horn Antennas	*Two horn antenna guides*	`https://opensourceradiotelescopes.org`
Loop Antenna	*Small loop antenna guide*	`https://opensourceradiotelescopes.org`
SatNOGS	*Global network of satellite ground-stations*	`https://satnogs.org`
Space Decentral	*Building a decentralized space program*	`https://spacedecentral.net`
Tiny Radio Telescope	*DIY radio telescope*	`https://hackaday.io/project/161556`
Ultrascope	*Automated robotics observatory*	`http://openspaceagency.com`
UPSat	*QB50 cubesat by the Libre Space Foundation*	`https://upsat.gr`

HOUSING

Wikihouse	*Open source house building designs*	`https://wikihouse.cc`
Open Building	*Open source furniture plans and more*	`https://openbuildinginstitute.org`
No Throw Design	*Downloadable furniture plans and instructions*	`https://nothrowdesign.com/you-make/`
Open Source Ecology	*Plans for building an entire village*	`https://opensourceecology.org/gvcs`
Divine on the Road	*Plans for building your dream van*	`https://divineontheroad.com/build-a-van`
The Vanual	*DIY Campervan Conversion*	`https://thevanual.com`
Obrary	*Furniture designs requiring CNC mill/laser*	`https://obrary.com`

COMMUNICATIONS

DAPNet	*Decentralized Amateur Paging Network*	`https://hampager.de`
Disaster Radio	*Solar powered, disaster-resistant network*	`https://disaster.radio`
HackRF	*Low cost open source SDR platform*	`https://github.com/mossmann/hackrf`
LibreCMC	*Embedded OS, supporting a range of routers*	`https://librecmc.org`
LimeSDR	*Open source software defined radio board*	`https://github.com/myriadrf`
Low-Tech Magazine	*How to build a solar powered website*	`https://solar.lowtechmagazine.com`
OpenVPN	*VPN software for secure data communications*	`https://openvpn.net`
OpenWRT	*Open Source wireless router firmware*	`https://openwrt.org`
Project Byzantium	*Linux-based emergency mesh networking*	`http://project-byzantium.org`
Subnodes	*Turning Raspberry Pi's into access points*	`http://subnodes.org`

COMPUTING

DLT one	*Modular, open hardware Linux tablet*	https://hackaday.io/project/164845
Libreboot	*Freedom-respecting boot firmware*	https://libreboot.org
MNT Reform	*Open source, modular laptop computer*	https://source.mntmn.com/MNT
OpenBook	*Open hardware e-reader*	https://hackaday.io/project/168761
Ploopy	*An open-source trackball*	https://ploopy.co

DRONES

ArduPilot	*Open source autopilot system*	http://ardupilot.org
Dronecode	*Open source platform for UAVs*	https://dronecode.org
Flone	*Complete plans for building a drone*	http://flone.cc/comprar-flone
Paparazzi UAV	*Open source drone autopilot systems*	http://paparazziuav.org

VIRTUAL & AUGMENTED REALITY

AtmosVR	*Open source XR headset*	https://hackaday.io/project/166006
HardlightVR	*Open source haptic feedback suit*	https://github.com/HardlightVR
HoloKit	*Low cost mixed reality platform*	https://holokit.io
OSVR	*The original open source VR headset*	http://osvr.org

VIRTUAL & AUGMENTED REALITY

Project North Star	*Leap Motion's AR headset*	github.com/leapmotion/ProjectNorthStar
Pupil	*Modular eyetracking platform*	https://pupil-labs.com/pupil
Relativity Headset	*VR Headset that costs ~$100*	https://relativty.net

AGRICULTURE

FarmBot	*Automated agriculture platform*	https://farm.bot
Food Computer	*Desktop food growing system*	http://openag.media.mit.edu/hardware
Farm Hack	*Various farming tool projects*	http://farmhack.org/tools
OSBeehives	*Open source beehive monitoring*	https://www.osbeehives.com

ELECTRICITY

OSSI	*Open source solar inverter project*	https://github.com/transistorgrab/OSSI
Green Optimistic	*Various generator design guides*	greenoptimistic.com/category/green-tech-2/how-to
Wind Generator	*55 watt 3D printed generator design*	https://hackaday.io/project/87345
Portal Point Generator	*Compact 100+ watt generator*	https://hackaday.io/project/159568

Continued →

MANUFACTURING / LASER CUTTING

LS-Laser	*Full guide for creating a CO2 laser cutter*	openbuilds.com/builds/ls-laser.7304
Lasersaur	*Build guide for Lasersaur cutter*	https://github.com/nortd/lasersaur/wiki
Laser V	*Smaller footprint laser engraver*	openbuilds.com/builds/much4-laserv-printed-version.937
LaseDuo	*Large, heavy duty laser cutter*	http://laserduo.com

MANUFACTURING / 3D SCANNING

MakerScanner	*Open source 3D laser scanner*	http://makerscanner.com
Ciclop	*Printable 3D scanner project*	https://github.com/bqlabs/ciclop
FabScan	*Raspberry Pi based laser scanner*	http://fabscan.org
FreeLSS	*3D printable / RPi-based scanner*	http://freelss.org

MANUFACTURING / 3D PRINTING

Creality Ender-3	*Hardware and software plans for Ender-3 printer*	github.com/Creality3DPrinting/Ender-3
Creality CR-10	*Hardware and software plans for CR-10 printer*	github.com/Creality3DPrinting/CR-10
Falla 3D	*Open source 3D printer that uses maglev system*	https://github.com/3dita/Falla3D
Indie i2	*Small footprint 3D printer*	openbuilds.com/builds/indie-i2.1976
Infinite 3D Printer	*Design for an automatic, conveyor 3D printer*	https://hackaday.io/project/114738

MANUFACTURING / 3D PRINTING

C-Bot	*Core XY style 3D printer with large print bed*	https://openbuilds.com/builds/c-bot.1146
3Drag	*3D printer which uses the RepRap philosophy*	https://reprap.org/wiki/3drag

MANUFACTURING / CNC MILLING

AE1 CNC	*Small footprint engraver/CNC*	github.com/Chris-Annin/AE1-CNC-engraver-router
C-beam Sphinx	*Large aluminium based CNC router*	openbuilds.com/builds/c-beam-sphinx.3605
Maslow CNC	*Community driven large format CNC*	https://maslowcnc.com
OpenBuilds MiniMill	*Small footprint desktop CNC*	openbuilds.com/builds/openbuilds-minimill.5087
OpenBuilds OX CNC	*Extremely detailed CNC build guide*	openbuilds.com/builds/openbuilds-ox-cnc-machine.341
Shapeoko	*The original open source desktop CNC*	https://wiki.shapeoko.com
X-Carve	*Popular modular CNC designs*	x-carve-instructions.inventables.com

MANUFACTURING / VACUUM FORMING

Vacuum Former	*Open source vacuum former build guide*	https://labs.tcbl.eu/projects/32

MANIFESTING REALITY

Thank you for taking the time to read through NODE Vol 02. I hope it gives you optimism about the future, and what I particularly want to ram home is the idea that it matters that you play an active role in not only consuming technology, but creating, and proliferating it.

Many people will try to convince you that politics, activism, placard waving, and the like are the only ways to 'change' things, but they're wrong. Technology is both the change agent, and the enabler of new ideas—this is how it's always been.

If you think about it, pretty much every single seismic change in human history came down to technology. From figuring out and sharing how to make pointy sticks for hunting, to creating paper, writing implements, metallurgy, printing presses, farming techniques, medicine, telephones, radios, trains, planes, computers, the Internet, spaceships, and everything in between.

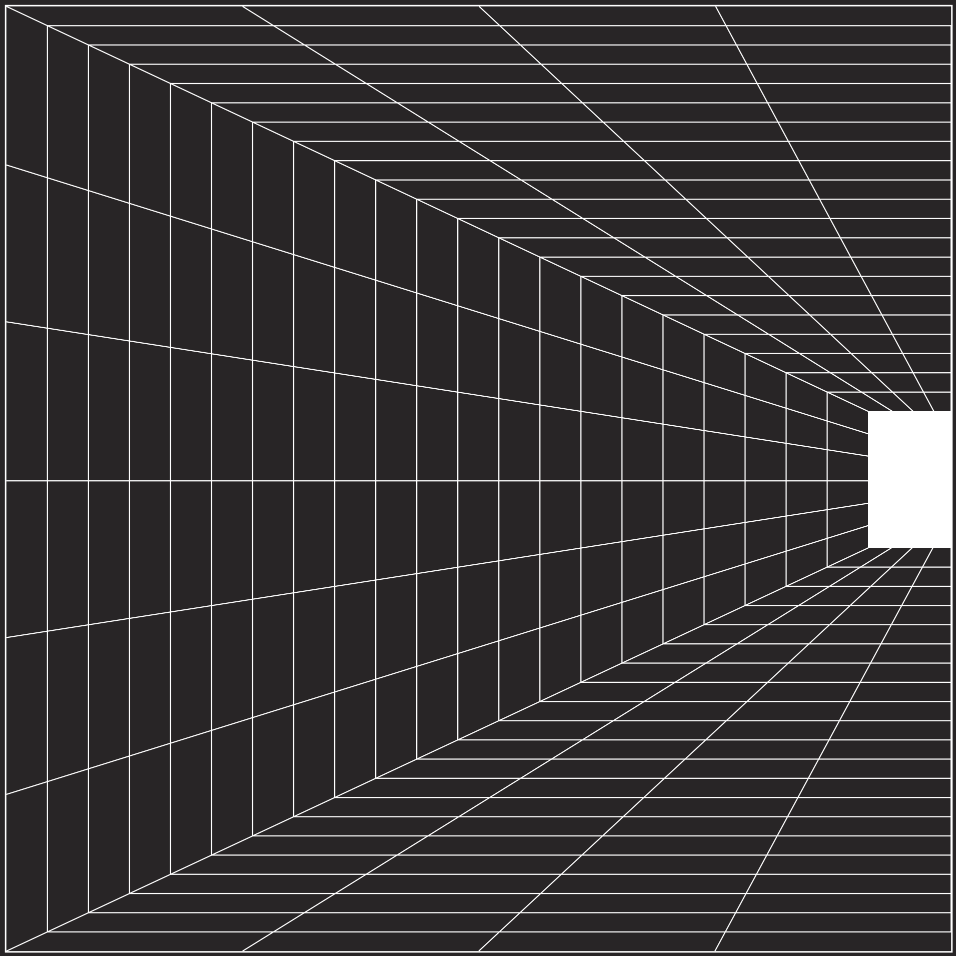
Like I mentioned in last issue’s "Satoshi Mindset" closing article, our civilization's heavy reliance on electronics means now more than any other time in history, each one of us has a chance to improve things on a small, or massive scale.

I know it's easy to make excuses. You don't have enough time, knowledge, money, connections, or whatever. I've told myself all those, many times, and you know what I found out? It's bullshit. The truth is, the best time to start is always NOW, using whatever you have at your disposal, even if that's nothing more than your own imagination.

Maybe you've seen a project you would like help out, or perhaps you've got your own ideas on how to solve a problem using technology. Either way, we're eager for you to succeed and share your findings with the world.

Use the hashtag *#nodevol02* somewhere on your project pages or social media. This will be our way to find each other across the web.

Keep building. The world needs you more than you know. Much love, NODE.



NODE VOL 02

Mike Dank, Editor and Writer (`editor@N-O-D-E.net`). Articles: p004, p030, p054, p086, p098, p106, p118, p158
NODE, Art Director and Writer (`mail@N-O-D-E.net`). Articles: p012, p018, p022, p028, p034, p038, p044, p048, p060, p078, p084, p092, p096, p098, p112, p124, p132, p134, p140, p148, p154, p166, p178

IMAGE CREDITS

p006 by USGS (Public Domain), p007 by Bell Telephone Magazine (1922), p010 by CIA (Public Domain), p019 p020 by Joey Castillo (CC-BY-SA 4.0), p024 p026 by Martti Malmi (CC-BY-SA 4.0), p039 p040 p041 p042 p043 by Apertus.org (CC-BY 4.0), p049 p050 p051 p052 p053 by Lukas F. Hartmann (CC-BY-SA), p055 p058 by Andre Staltz (CC-BY 4.0), p061 p062 p063 by Glia Project (CC-BY-SA 4.0), p064 p065 by Frankie Flood (CC-BY-SA 4.0), p066 p067 by Indi.Global (CC-BY-SA 4.0), p068 p069 by Joseph Xu/Michigan Engineering, Communications & Marketing (CC-BY-SA 4.0), p070 left by Joel Collins / Open Flexure (CC-BY 4.0), p070 right by Dan Berard (CC-BY-SA 4.0), p071 left by Open Source Imaging (CC-BY-SA 4.0), p071 right by OpenPCR (CC-BY-SA 4.0), p072 p0073 by Precious Plastic (CC-BY-SA 4.0), p074 p075 p076 by Tympan (CC-BY-SA 4.0), p099 p101 p104 by Niamfrifruli (CC-BY-SA 4.0), p103 by July an88 (CC-BY-SA 4.0), p107 by BJ-AKI (Pixabay License), p111 by dokumol (Pixabay License), p149 p150 by The AKASHA Foundation (CC-BY-SA 4.0).

THANK YOU

Nicolas Pace, Mihai Alisie, Joshua Long, Martti Malmi, Lukas F. Hartmann, Andre Staltz, Frankie Flood, Tarek Loubani, Tamas Kocsis, Joey Castillo, Lukas Winter, Addie Wagenknecht, Joel Murphy, Elias Jaffa, Dan Berard, Joel Collins, Dan Newman, Elliot Rouse, Josh Perfetto, Richard Bowman, Fernanda Zapata-Murrieta, Burak Nehbit, Sam Patterson, Jeremy Kaufman, Sebastian Pichelhofer, RexOr, Esteban Ordano, Matthew Hodgson, Paul Frazee, Shawn Wilkinson, Manuel Moritz, & Baptiste.